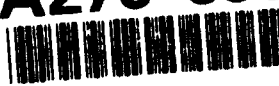




2

**AD-A275 398**  


**DTIC**  
ELECTRONIC  
FEB 8 1994  
  


# **PROCESS ENGINEERING WITH THE EVOLUTIONARY SPIRAL PROCESS MODEL**

**SPC-93098-CMC**

**VERSION 01.00.06**

**JANUARY 1994**

**EXHIBIT A**  
Approved for public release  
Distribution Unlimited

**94-04212**  


**94 2 07 05 6**

# PROCESS ENGINEERING WITH THE EVOLUTIONARY SPIRAL PROCESS MODEL

DTIC QUALITY INSPECTED 8

SPC-93098-CMC

VERSION 01.00.06

JANUARY 1994

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By <i>Per A258966</i>	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
<i>A-1</i>	

Produced by the  
SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION  
under contract to the  
VIRGINIA CENTER OF EXCELLENCE  
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER

SPC Building  
2214 Rock Hill Road  
Herndon, Virginia 22070

Copyright © 1994, Software Productivity Consortium Services Corporation, Herndon, Virginia. Permission to use, copy, modify, and distribute this material for any purpose and without fee is hereby granted consistent with 48 CFR 227 and 252, and provided that the above copyright notice appears in all copies and that both this copyright notice and this permission notice appear in supporting documentation. This material is based in part upon work sponsored by the Advanced Research Projects Agency under Grant #MDA972-92-J-1018. The content does not necessarily reflect the position or the policy of the U. S. Government, and no official endorsement should be inferred. The name Software Productivity Consortium shall not be used in advertising or publicity pertaining to this material or otherwise without the prior written permission of Software Productivity Consortium, Inc. SOFTWARE PRODUCTIVITY CONSORTIUM, INC. AND SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION MAKE NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS MATERIAL FOR ANY PURPOSE OR ABOUT ANY OTHER MATTER, AND THIS MATERIAL IS PROVIDED WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND.

---

**ADW is a registered trademark of KnowledgeWare, Inc.**

**Ernst & Young Navigator Systems Series is a service mark of Ernst & Young International, Ltd.**

**Microsoft Project is a trademark of Microsoft Corporation.**

# CONTENTS

<b>PREFACE .....</b>	<b>xi</b>
<b>ACKNOWLEDGMENTS .....</b>	<b>xiii</b>
<b>1. INTRODUCTION .....</b>	<b>1-1</b>
1.1 Overview .....	1-1
1.2 Purpose and Scope .....	1-1
1.3 Intended Audience .....	1-2
1.4 Typographic Conventions .....	1-3
<b>2. PROCESS ENGINEERING CONCEPTS AND ISSUES .....</b>	<b>2-1</b>
2.1 Overview .....	2-1
2.2 Process Life Cycle .....	2-1
2.3 Process Engineering Issues .....	2-2
2.3.1 High-Level Process Definitions .....	2-3
2.3.2 Low-Level Process Definitions .....	2-4
2.3.3 Process Definitions That Mandate a Life Cycle .....	2-5
2.4 Emerging Process Engineering Concepts .....	2-5
2.4.1 Process Assets .....	2-6
2.4.1.1 Standard Process Architecture .....	2-7
2.4.1.2 Specified Process Steps .....	2-8
2.4.1.3 Product Life-Cycle Model Descriptions .....	2-9
2.4.1.4 Standard Process Definition .....	2-9
2.4.1.5 Project Process Architecture .....	2-10
2.4.1.6 Project Process Definition .....	2-10



2.4.1.7 Process Measures Database .....	2-10
2.4.2 Process Engineering Process .....	2-10
2.4.2.1 Architect Standard Process .....	2-11
2.4.2.2 Design Standard Process .....	2-13
2.4.2.3 Define Standard Process .....	2-13
2.4.2.4 Tailor Project Process .....	2-14
2.4.2.5 Instantiate Project Process .....	2-15
2.4.2.6 Provide Continuous Improvement .....	2-15
2.5 Summary .....	2-16
<b>3. EXTENDING THE EVOLUTIONARY SPIRAL PROCESS MODEL TO PROCESS ENGINEERING .....</b>	<b>3-1</b>
3.1 Overview .....	3-1
3.2 Evolutionary Spiral Process Model Overview .....	3-1
3.2.1 Step 1: Understand Context .....	3-2
3.2.2 Step 2: Analyze Risks .....	3-3
3.2.3 Step 3: Plan Development .....	3-4
3.2.4 Step 4: Develop Product .....	3-4
3.2.5 Step 5: Manage and Plan .....	3-5
3.3 Process Engineering With the Evolutionary Spiral Process Model .....	3-5
3.4 Summary .....	3-7
<b>4. ENGINEERING STANDARD PROCESS ASSETS .....</b>	<b>4-1</b>
4.1 Overview .....	4-1
4.2 Engineering a Standard Process .....	4-1
4.2.1 Step 1: Understand Context .....	4-2
4.2.2 Step 2: Analyze Risk .....	4-4
4.2.3 Step 3: Plan Asset Development .....	4-5
4.2.4 Step 4: Develop Process Assets .....	4-7

4.2.4.1 Process Capability Baseline .....	4-8
4.2.4.2 Standard Process Architecture .....	4-9
4.2.4.3 Specified Process Steps .....	4-9
4.2.4.4 Product Life-Cycle Models .....	4-11
4.2.4.5 Standard Process Definition .....	4-11
4.2.4.6 Process Measures Database .....	4-12
4.2.5 Step 5: Planning and Management .....	4-12
4.3 Optimizing Organizational Process Assets .....	4-13
4.4 Summary .....	4-15
<b>5. PROJECT PROCESS ENGINEERING .....</b>	<b>5-1</b>
5.1 Overview .....	5-1
5.2 Documenting the Project Process in the First Cycles .....	5-3
5.2.1 Identify and Evaluate Project Process Drivers .....	5-3
5.2.2 Analyze Project Process Risks and Plan for Risk Aversion Strategies .....	5-4
5.2.3 Planning for the Development of the Project Process Definition .....	5-5
5.2.4 Developing the Project Process Definition .....	5-5
5.2.5 Baselining the Project Process Definition and Committing to Proceed .....	5-6
5.3 Tailoring and Instantiating the Project Process Definition for a Cycle .....	5-7
5.3.1 Determine Cycle Process Drivers and Alternatives .....	5-7
5.3.2 Analyze Cycle Process Risks and Plan for Risk Aversion Strategies .....	5-8
5.3.3 Document the Cycle Process .....	5-8
5.4 Enacting the Cycle Process .....	5-10
5.5 Improving the Project Process .....	5-11
5.6 Summary .....	5-12
<b>APPENDIX A. EVOLUTIONARY SPIRAL PROCESS ACTIVITY SPECIFICATIONS .....</b>	<b>A-1</b>
A.1 Overview .....	A-1

A.2 Define Approach .....	A-5
A.3 Develop/Update Estimate of the Situation .....	A-6
A.4 Review Context .....	A-8
A.5 Perform Risk Analysis .....	A-10
A.6 Review Risk Analysis .....	A-13
A.7 Plan Risk Aversion .....	A-14
A.8 Commit to Aversion Strategy .....	A-15
A.9 Execute Risk Aversion .....	A-17
A.10 Review Alternative .....	A-18
A.11 Plan and Schedule .....	A-19
A.12 Commit to Plan .....	A-22
A.13 Develop and Verify Product .....	A-23
A.14 Monitor and Review .....	A-25
A.15 Review Technical Product .....	A-27
A.16 Product Change Control .....	A-28
A.17 Review Progress .....	A-29
A.18 Update Spiral Planning Documents .....	A-30
A.19 Commit to Proceed .....	A-32
<b>APPENDIX B. PRODUCT DEVELOPMENT ACTIVITY SPECIFICATIONS ...</b>	<b>B-1</b>
B.1 Software Systems Engineering .....	B-1
B.1.1 Specify Operational Concept .....	B-1
B.1.2 Formulate Potential Approaches .....	B-3
B.1.3 Define System Requirements .....	B-5
B.1.4 Develop System Architecture .....	B-8
B.1.5 Design User Interface .....	B-13
B.1.6 Analyze System Performance .....	B-16
B.1.7 Analyze System Dependability .....	B-18

B.1.8 Analyze System Reusability .....	B-20
B.1.9 Analyze System Modifiability .....	B-22
B.1.10 Analyze System Functionality .....	B-24
B.1.11 Recommend Software System Development Strategy .....	B-26
B.1.12 Integrate Software System Components .....	B-28
B.2 Software Engineering .....	B-29
B.2.1 Define Software and Interface Requirements .....	B-29
B.2.2 Design Software Architecture .....	B-32
B.2.3 Analyze Software Performance .....	B-35
B.2.4 Analyze Software Reusability .....	B-37
B.2.5 Analyze Software Modifiability .....	B-39
B.2.6 Analyze Software Functionality .....	B-41
B.2.7 Design Database .....	B-43
B.2.8 Design Software Components .....	B-45
B.2.9 Create Software Components .....	B-47
B.2.10 Integrate Software Components .....	B-49
B.3 Product Verification and Validation .....	B-50
B.3.1 Plan Verification and Validation .....	B-50
B.4 System Verification and Validation .....	B-51
B.4.1 Validate System Requirements .....	B-51
B.4.2 Define System Integration Verification .....	B-52
B.4.3 Verify System Integration .....	B-54
B.4.4 Demonstrate System Capabilities .....	B-55
B.5 Software Verification and Validation .....	B-56
B.5.1 Validate Software Requirements .....	B-56
B.5.2 Define Software Component Verification .....	B-57
B.5.3 Verify Software Component .....	B-59

B.5.4 Define Software Integration Verification .....	B-60
B.5.5 Verify Software Integration .....	B-62
B.6 Operation and Maintenance .....	B-63
B.6.1 Plan System Installation .....	B-63
B.6.2 Install System .....	B-65
B.6.3 Provide Operational Support .....	B-66
B.6.4 Identify New Operational Capabilities .....	B-68
<b>APPENDIX C. ARTIFACT DESCRIPTIONS .....</b>	<b>C-1</b>
<b>LIST OF ABBREVIATIONS AND ACRONYMS .....</b>	<b>Abb-1</b>
<b>GLOSSARY .....</b>	<b>Glo-1</b>
<b>REFERENCES .....</b>	<b>Ref-1</b>
<b>BIBLIOGRAPHY .....</b>	<b>Bib-1</b>
<b>INDEX .....</b>	<b>Ind-1</b>

## FIGURES

Figure P-1. Structure for Integrated Application of Consortium Technologies .....	xi
Figure 2-1. A Typical Process Life Cycle .....	2-2
Figure 2-2. Contents of the Process Asset Library .....	2-7
Figure 2-3. Product Life-Cycle Model .....	2-9
Figure 2-4. Project Process Architecture .....	2-10
Figure 2-5. A Modified View of the Process Life Cycle .....	2-12
Figure 3-1. The Conceptual Evolutionary Spiral Process Model: A Management Process .	3-3
Figure 3-2. Process Engineering With the Evolutionary Spiral Process Model .....	3-6
Figure 4-1. Standard Process Engineering Process .....	4-3
Figure 4-2. Synthesis Process Model .....	4-15
Figure 5-1. Project Process Engineering .....	5-1
Figure A-1. Entry-Task-Validation-eXit Notation .....	A-2

## TABLES

Table P-1. Consortium Guidebooks and Related Practices .....	xii
Table A-1. Evolutionary Spiral Process Activities .....	A-1

## PREFACE

The technology described in this guidebook is part of a broad approach to software productivity improvement. This preface provides an overview of that approach and identifies the series of guidebooks that support it. These guidebooks were developed by the Software Productivity Consortium under contract to the Virginia Center of Excellence for Software Reuse and Technology Transfer (VCOE). For a complete listing of VCOE guidebooks and products, call the Software Productivity Consortium's Technology Transfer Clearinghouse at (703) 742-7211.

Each technology has been packaged so it can be used without reference to the other technologies. However, it is also possible to combine these technologies into an integrated approach for product development. Figure P-1 shows how the guidebooks for these technologies relate to the practices of software development organizations.

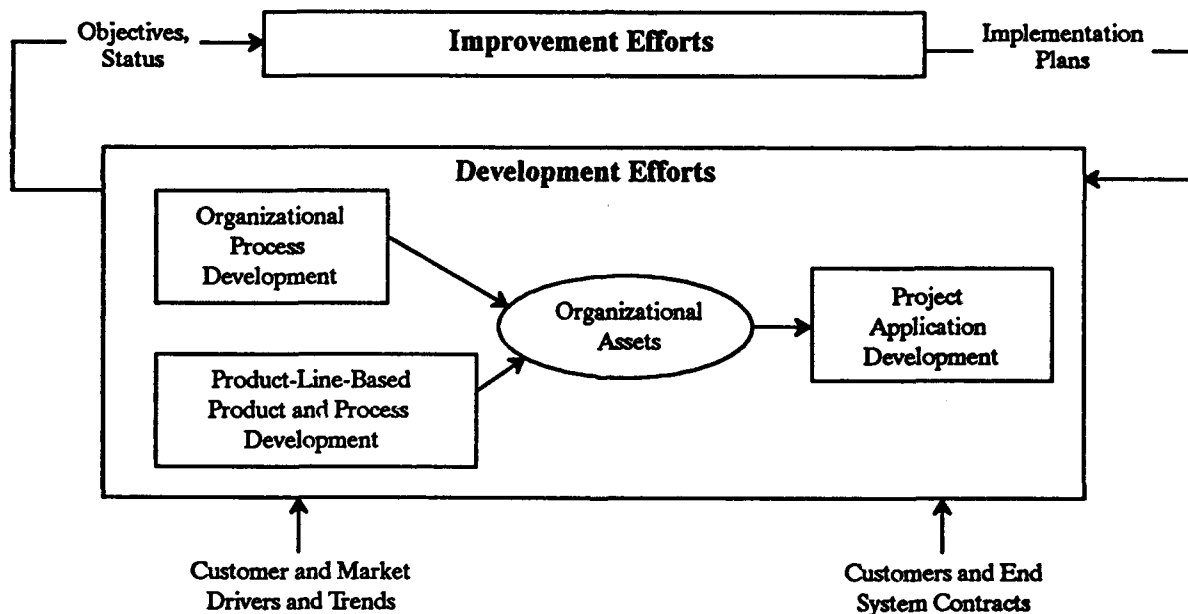


Figure P-1. Structure for Integrated Application of Consortium Technologies

These practices are composed of:

- **Improvement Efforts (IE).** Application of technology to improve software development efforts. These efforts require managed approaches to assessment of objectives and current capabilities, planning for the improvement, implementation of the plan, and measurement of success.
- **Development Efforts.** Development of products that meet the needs of customers and markets or products that make the organization more competitive in meeting expected future needs.



- **Organizational Process Development (OPD).** Development of standardized organizational process assets (e.g., process and method descriptions, process enactment tools) tailored for a particular organization.
- **Product-Line-Based Product and Process Development (PLD).** Development of integrated product and process assets (e.g., core products and processes for adapting them for particular customer needs) appropriate for a particular product line.
- **Project Application Development (PAD).** The tailoring and application of organizational assets for a particular product development effort.

Table P-1 describes how existing products can be integrated to address your organizational practice.

Table P-1. Consortium Guidebooks and Related Practices

Guidebook	Part Number	Relationship to Software Practice
<i>Consortium Requirements Engineering Guidebook</i>	SPC-92060-CMC	Used for defining and analyzing requirements in PAD. Adaptable for use in PLD.
<i>Managing Process Improvement: A Guidebook for Implementing Change</i>	SPC-93105-CMC	Supports IE by providing a process and supporting guidance for initiating and maintaining an organizational process improvement program.
<i>Process Definition and Modeling Guidebook</i>	SPC-92041-CMC	Provides methods for defining and documenting processes so they can be analyzed, modified, and enacted. Supports IE and OPD.
<i>Process Engineering With the Evolutionary Spiral Process Model</i>	SPC-93098-CMC	Used to iteratively plan, manage, and control PAD and PLD. Used to construct organization-specific processes in PLD and tailor them in PAD.
<i>Reuse Adoption Guidebook</i>	SPC-92051-CMC	Supports IE by providing specific process improvement activities for incorporating reuse practices.
<i>Reuse-Driven Software Processes Guidebook</i>	SPC-92019-CMC	Provides development approaches for PLD (domain engineering) and PAD (application engineering) of reusable software assets.
<i>Software Measurement Guidebook</i>	SPC-91060-CMC	Supports IE by providing methods for quantitative assessment of project status.
<i>Using New Technologies: A Technology Transfer Guidebook</i>	SPC-92046-CMC	Supports IE by providing a process that addresses how to get an organization to use new technologies.

## **ACKNOWLEDGMENTS**

The primary authors for this guidebook were Kirsten Blakemore, Donna Garfield, and Jim Marple, and contributing authors were Robert Hofkin, Ph.D.; Tim Powell; Sue Rose; and Patricia Remacle. However, many people have aided in the creation of the ESP technology since its inception in 1991.

- The first two versions of the Evolutionary Spiral Process (ESP) model technology were created by John Blyskal; Ted Davis; Mary Eward; Robert Hofkin, Ph.D.; Tim Powell; and Patricia Remacle.
- The first two versions of the ESP model technology were formally validated by Guy Cox, Ted Davis, Fred Hills, and Roger Williams.
- David Nettles has continually provided overall direction and guidance.

In addition, this guidebook uses and makes reference to several existing Consortium technologies.

- Synthesis technology was created by Grady Campbell, Jim O'Connor, Neil Burkhard, Jeff Facemire, and Rich McCabe.
- Software measurement guidance was authored by Robert Cruickshank, John Gaffney, and Richard Werling.
- Formal process definition and modeling technology was created by Richard Bechtold and Robert Lai.
- Software engineering technology transfer guidance was authored by John Christian, Ph.D.; Mary Eward; Sam Redwine; and Louis Tornatzky, Ph.D.

The review and comments of Richard Bechtold, Jim Blake, Tim Powell, Sam Redwine, and Roger Williams greatly aided in the development of this guidebook.

The coordination, editing, and processing of this document was performed by the Environment and Support Services Division.

*This page intentionally left blank.*

# **1. INTRODUCTION**

## **1.1 OVERVIEW**

Imagine a standard, well-defined, and rigorously optimized process for constructing buildings. The activities in the construction process all have standard definitions, are always sequenced in the same way, and never vary in the time and resources needed to perform them. Type and quantity of material, as well as the number and skill mix of architects, engineers, and contractors, are known precisely. Standard tools and equipment are always used. Site descriptions, floor plans, blueprints, and other design and production documents never vary to avoid changes to the standard process.

This construction process tends to produce square, concrete structures with the same dimensions, the same floor plan, and the same landscaping—well-built, reasonably priced, and quickly constructed, but of questionable value to customers who have unique requirements or constraints.

Differences in underlying culture, natural settings, and human-imposed expectations, decisions, and limitations are some of the reasons why construction processes may vary in slight to significant ways (Snyder and Catanese 1979). Such motivating factors, or process drivers, also drive software development. For example, following a standard, unvarying process for developing software may quickly produce error-free and low-cost software, but the resulting software product(s) may not satisfy differences in organizational and client cultures; human-imposed objectives, alternatives, and constraints; and preexisting knowledge or assets.

The industry is beginning to recognize the importance of standard and defined development processes to producing consistently high-quality software within budget and on schedule. However, standard processes must be engineered to accommodate unique project characteristics. This guidebook is a first step toward introducing a framework for engineering processes that combines the advantages of process standardization with the flexibility to address unique process drivers.

## **1.2 PURPOSE AND SCOPE**

This guidebook describes how process engineering helps to produce the right development process for an organization and/or project and shows how the Evolutionary Spiral Process (ESP) model can be extended to support process engineering steps. A development process is the set of steps, or activities, for developing a software product and its supporting products. Process engineering refers to the specific actions that attempt to generate a quality software development process, given organizational and/or project-unique process drivers.

The guidebook offers expanded, though not yet complete, process engineering guidance at both the organizational and project levels. Information in this guidebook is at a first level of maturity; that is, it synthesizes the results of initial research and theory and will be matured in future versions as the

material is taught and used, lessons are learned, and the concepts evolved. In addition, these initial concepts will be further elaborated as confidence in the foundation material increases.

This version of the guidebook covers the following topics:

- Section 1 describes the guidebook in terms of its purpose, scope, intended audience, and typographic conventions.
- Section 2 describes basic process engineering concepts and issues, including process assets and process life-cycle steps.
- Section 3 provides an overview of the ESP model and shows how the model can be extended to process engineering.
- Section 4 provides a discussion on engineering standard process assets.
- Section 5 provides a discussion on process engineering at the project level.
- Appendix A presents the ESP model activity specifications.
- Appendix B provides some generic product development activity specifications that can be used when engineering standard process assets or project-specific process definitions.
- Appendix C contains a list of product development artifact descriptions.
- The List of Abbreviations and Acronyms contains abbreviations and acronyms used in this guidebook and their definitions.
- The Glossary contains a list of terms used in this guidebook and their definitions.
- The References section contains sources of information used in this guidebook.
- The Bibliography contains additional sources of information.

Because the industry has not yet set standards for concepts and definitions, it is recommended that the sections of this guidebook be read in order. Otherwise, you may be confused by concepts presented, defined, or used in a way that is different from other sources.

### 1.3 INTENDED AUDIENCE

This guidebook is intended for use by the following audiences:

- A software engineering process group (SEPG) member, or other process improvement specialist, interested in defining an organizational process definition and tailoring it for use on a specific project.
- A project manager interested in using the ESP model to define a software development process that best meets the needs and requirements of the project.

## 1.4 TYPOGRAPHIC CONVENTIONS

This guidebook uses the following typographic conventions:

**Serif font** ..... General presentation of information.

**Serif font, initial capitalization** ..... Names of processes and activities in the model.


**Boldfaced serif font** ..... Section headings and emphasis.

*Italicized serif font* ..... Publication titles.

***Boldfaced italicized serif font*** ..... Run-in headings in bulleted lists and low-level titles in the process sections of guidebooks.

In this guidebook, figures that depict a process use the following symbolism:

 ..... Activity or step.

 ..... X produces product or work product.

 ..... X uses product or work product.

 ..... Work flow reiterates work product development.

*This page intentionally left blank.*

## 2. PROCESS ENGINEERING CONCEPTS AND ISSUES

The systems built today are just too complex for the mind of man to foresee all the ramifications purely by the exercise of the analytical imagination.

*Report of the Defense Science Board Task Force on  
Military Software*

### 2.1 OVERVIEW

Many process groups today are faced with the difficult problem of defining the product development process for a large, complex, and highly variable organization, as well as for the multiple and diverse projects within that organization. Finding the right solution requires structure and discipline, including careful analysis of the problem, designing an architectural framework within which the problem can be decomposed, and decomposing the problem into smaller, more manageable pieces.

For example, imagine trying to build a large, complex software system without first designing a software architecture: interface problems are prevalent, requirements are implemented inconsistently or not at all, and design constraints are violated. Like software engineering, process engineering should also represent complex interactions among process steps through a high-level process architecture that can be subsequently refined and elaborated. Process engineering should follow a structured approach that elaborates the process through different forms of representation and level of detail until an enactable or performable level is reached.

This section introduces process engineering in terms of a typical process life-cycle model, discusses some difficulties the industry is discovering as it attempts to engineer and enact process definitions, and presents some emerging process engineering concepts that begin to address these issues.

### 2.2 PROCESS LIFE CYCLE

Process is a partially ordered set of steps intended to accomplish specified objectives; process engineering refers to the construction of that process. This guidebook constrains the definition of process engineering to refer to product development objectives only. Process engineering constructs the best possible process for developing a system or software product(s).

Figure 2-1 shows a typical process life-cycle model. This process life-cycle model provides an initial process engineering framework. Unlike product life cycles, such as the traditional waterfall, the life cycle of a process should continuously improve and evolve, at least until the industry experiences a radical paradigm shift in the way it develops software and software-intensive system products. Each of the process life-cycle phases is discussed briefly below.



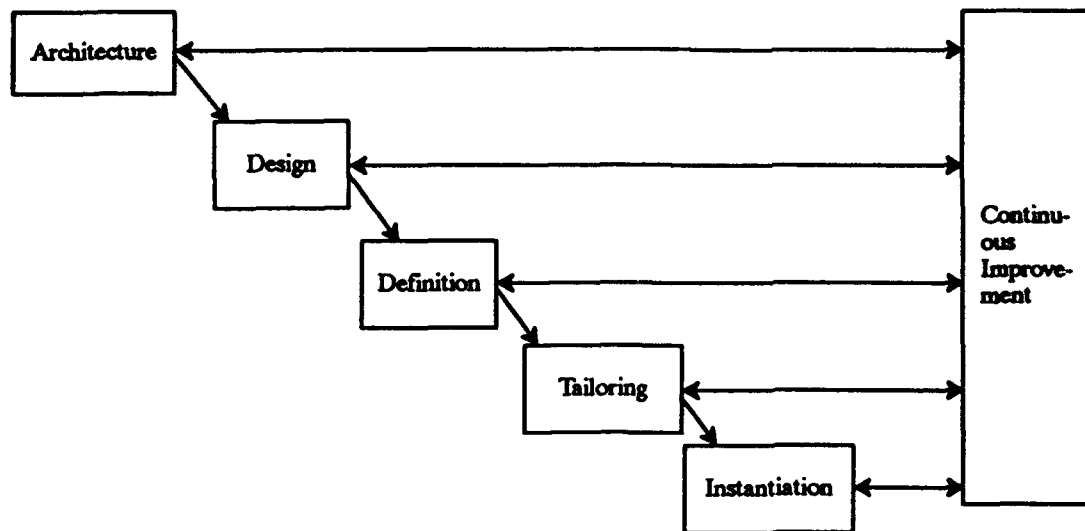


Figure 2-1. A Typical Process Life Cycle

- **Architecture.** This phase involves defining organizational context, process standards, and the major process steps that will be modeled within the organization and how those steps interact.
- **Design.** This phase involves defining one or more product life-cycle models that the process will support and how the life cycle model(s) integrate with the conceptual framework defined in the architecture stage.
- **Definition.** This phase involves decomposing and specifying the internal structure of each of the process steps in the process architecture.
- **Tailoring.** This phase involves selecting the product life cycle that will be used on a specific project and tailoring the process step specifications to accomplish each product life-cycle stage.
- **Instantiation.** This phase creates an enactable or performable process for a specific project by binding the process definition to resources.
- **Evolution.** This is a continuous phase that corrects known problems or evolves the process to meet new needs.

In the past few years, several SEPGs have followed the traditional process life cycle shown in Figure 2-1 or a similar model. The result has generally been a single, comprehensive organizational process definition detailed to the enactable level that reflects the various policies and procedures used throughout the organization. As discussed in the Section 2.3, this linear approach to process engineering and the resulting comprehensive process definition have resulted in significant issues.

## 2.3 PROCESS ENGINEERING ISSUES

In the last several years, the industry started to approach development of software and software-intensive systems no longer as a black art but as an engineering discipline that should be used in a consistent and repeatable manner. The industry is developing process definitions to document management, development, and support steps and procedures for developing a system or software

product. All the projects across the organization are meant to use the process definitions. To actually enact the development process, a project needs a well-defined and comprehensive set of information, including:

- The set of activities necessary to develop software and its supporting products
- How the activities relate to each other
- A description of each activity
- When to start and stop each activity
- What inputs each activity needs
- What each activity accomplishes or produces
- Activity cost, schedule, and staffing estimates
- What personnel, methods, practices, and tools will be used to perform each activity

As stated by Feiler and Humphrey (1992), process definitions will become valued and used only to the extent that they make high-quality software easier and more economical to produce. To achieve this, process definitions must be both useful to the practitioners and reasonably economical to produce. Experience to date, however, demonstrates that the development of a comprehensive process definition can be very expensive and time consuming.

In many cases, a process group has taken several calendar years and thousands of labor hours to produce organizational process definitions that development projects are just beginning to use. Often, these descriptions simply do not correspond to the processes actually performed during software development or maintenance (Curtis, Kellner, and Over 1992). As process engineers analyze the results of projects using organizational standard process definitions for the first time, the disconnect between the process definition and how a project actually performs becomes apparent. Some of the issues that contribute to this disconnect include the following:

- If the organizational process definition is developed at too high a level of detail, it may fail to provide sufficient guidance to the project.
- If an organizational process definition is defined at too low a level of detail, it may be difficult to adapt to accommodate project-specific objectives and constraints.
- If an organizational process definition is closely tied to a specific life-cycle model, it may break down when a project attempts to use a life-cycle variant or an alternate life cycle.

Sections 2.3.1 through 2.3.3 discuss each problem further.

### **2.3.1 HIGH-LEVEL PROCESS DEFINITIONS**

Process definitions that you represent too abstractly can result when you model an organizational process using large-grained activities or provide an incomplete set of information. If you break down a process definition into large-grained activities, the activity descriptions do not provide enough

detail. The activities may decompose into subactivities with no explicit information concerning the sequencing or dependencies between the subactivities. Also, the dependency of a particular subactivity on a particular input of the parent activity is not defined. An example is a parent activity containing a list of various roles, but how each role contributes to the completion of the subactivities is not specified. Enactment of these high-level activities will result in ad hoc practices under the guise of a common organizational process definition (Krasner et al. 1992).

Traditionally, the organizational process definition models a functional process perspective, that is, what activities need to be done and how they are sequenced. However, you must integrate many forms of information to adequately describe the software process. Information that agents normally need to enact the process also includes:

- What is going to be done?
- Who is going to do it?
- When and where will it be done?
- How will it be done?
- Who depends on it being done?
- How does someone measure progress?
- How should someone document results?
- How should someone verify results?

Normally, traditional activity-based process definitions do not adequately address all of these questions. For instance, the process roles do not explicitly map to organizational functions or the format, verification criteria, and relationship between artifacts do not have explicit definitions.

### 2.3.2 LOW-LEVEL PROCESS DEFINITIONS

Rather than having a single monolithic process that all projects must use, organizations will likely find that different projects will have differing needs (Feiler and Humphrey 1992). Unfortunately, it is not unusual for an organization to spend a significant amount of internal resources and calendar time to develop more than a thousand pages of process definition at the lowest levels of detail. Although these low-level definitions are more enactable and, thus, more easily automated, they are difficult to tailor to projects with differing objectives and constraints.

There seems to be a trend whereby an organization assesses its process maturity using the Software Process Assessment method (Paulk et al. 1993) or another technique and immediately launches into a concentrated effort to develop an elaborate process definition that is seen as the ideal way to conduct business, instead of reflecting what is currently done in the organization. The result is often a large, tightly integrated, and detailed set of documentation that is difficult to absorb at the project level or to adapt to address project-specific objectives and constraints.

Because a process definition may be for an organization, a class of projects, a specific project team, or an individual professional, it is difficult to gauge what the appropriate level of elaboration might

be. Fully elaborated process definitions are complete or fit for enactment. Completeness, however, depends on context because a definition that is complete for one process agent may not be for another. For example, to a systems engineer, there may be no need to decompose the process step Develop Software System any further, yet the software engineer obviously needs to refine this process step to a much lower level of detail.

### 2.3.3 PROCESS DEFINITIONS THAT MANDATE A LIFE CYCLE

A product life cycle defines the key states that a product passes through as it matures over its useful life. Most development process models to date have been so tied to a specific life-cycle model that it is very difficult to distinguish the process from the life cycle. As stated in Curtis, Kellner, and Over (1992), the industry has traditionally considered these life-cycle representations as process models. However, visibility into the life cycle is important but does not identify all of the process steps needed to reach the life-cycle states and is, therefore, an incomplete process model at best. Additionally, many of today's life-cycle models do not account for the issue of perspective and simply depict the highest level of elaboration in their representation.

Software process definitions that are built on top of conventional life-cycle models are generally grounded in the assumption that successful software is developed in a lock-step procession of specification, design, code, and test. A flexible development process allows for the selection of different intermediate products, milestones, and activities for different projects. Process definitions that are built upon life-cycle models usually focus on product development and fail to show the many elemental process building blocks necessary for managing and coordinating a project (Curtis, Kellner, and Over 1992).

Instead of building the process framework around a specific life cycle, the process definition should be adaptable to support multiple development paradigms, such as prototyping, operational specification, and transformational implementation, where the life cycle of the product may not be readily apparent at the beginning of the project but may evolve as the project progresses. This approach to a flexible development process allows project managers to decide what constitutes progress in each unique project situation (Agresti 1986). You should incorporate guidance into the process definition on how to generate an appropriate life cycle for the product being produced and select the process steps that will best reach each of the life-cycle states.

## 2.4 EMERGING PROCESS ENGINEERING CONCEPTS

To a large extent, the issues that Section 2.3 describes are symptoms of the difficulties you currently encounter when attempting to tailor a standard process definition for a specific project. Part of the problem is that there are two different perspectives of the term "tailor" that process groups generally do not consider when chartered with process engineering activities.

For example, consider what tailoring means in the context of buying a suit. A customer can go to a department store and select a suit from the rack that is approximately the desired color, style, and size. The suit is then slightly tailored to more closely fit specific measurements, and is generally completed within the week.

The other option is to go to a tailor, who has everything needed to construct a suit according to customer specifications, including material, threads, sewing equipment, etc. The suit will very likely be more expensive than the one purchased in the department store and will generally take longer to complete.

Obviously, it is quicker and more cost effective to go to the department store and buy a suit. However, this observation is true because department stores do not stock only navy blue suits in size 42 long that are tailored to fit every customer. Rather, suits in several colors, sizes, and styles are available so that a customer can choose something approximately close to his needs that can then be quickly tailored to a more perfect fit.

Many organizations have expended significant time and resources developing the navy blue, size 42 long (process definition) and are currently struggling to tailor it to every project regardless of unique needs. Perhaps a better solution is for an organization to consider one or a combination of the following strategies:

- Develop whatever is needed for constructing a project-level process based on unique specifications. After you construct and enact a project process, evaluate, categorize, and “hang it on the rack” to build up a selection of key process definitions that you can slightly tailor and reuse.
- Develop a selection of standard process definitions based on business areas, which you can define at a low level of detail because of visibility into the product line within each business area. You can select an existing standard process definition that approximately addresses project-unique process drivers, and slightly tailor it.

To implement either of these strategies, it is helpful to:

- Consider the development of reusable process assets or anything that is useful in the engineering and enactment of processes that you can reuse in some manner.
- Consider different levels of process drivers or key characteristics that place requirements on the process.

Section 2.4.1 discusses the process assets concept and describes an approach for defining and elaborating process assets through different levels of detail based on unique process drivers.

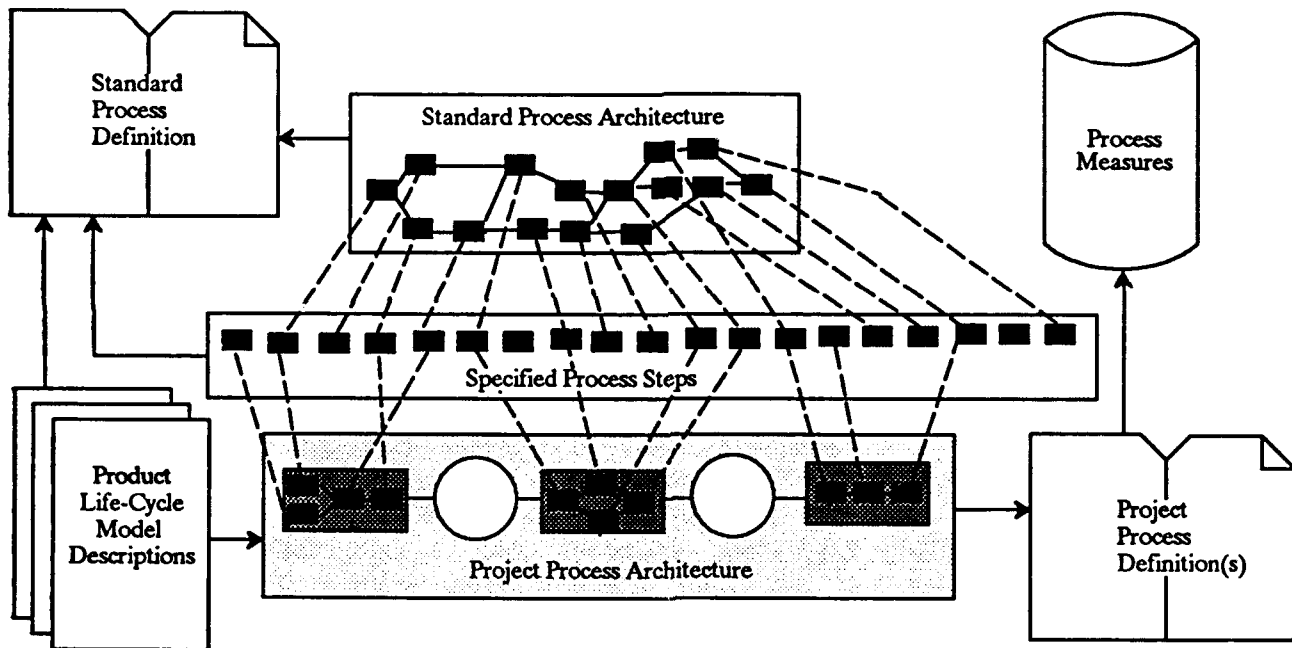
### **2.4.1 PROCESS ASSETS**

The concept of developing a repository of reusable process assets to help leverage the use of defined processes into widespread practice was an area that the Software Technology for Adaptable Reliable Systems (STARS) and the Software Engineering Institute (SEI) recently examined. The motivation for the work was to provide a starting baseline for software organizations that lack a formal, defined process (Kasunic et al. 1992). Another advantage of the process asset concept is that it begins to move the industry away from its tendency to define a single, fully elaborated, voluminous, and inflexible process definition and into an environment where you develop and maintain reusable assets at an organizational level and then select, tailor, and integrate them based on project needs.

Process reuse is analogous to software reuse. The process reuse concept is that you need not always define processes from scratch and that national, organizational, or local libraries may make available

previous instances of process assets. Process assets can be public or proprietary and can be in many forms, such as libraries, domain-specific development environments, project management tools, etc. Process assets can be considered as anything that is useful in the engineering and enactment of processes that you can reuse in some manner and can be documentation that resides permanently on a shelf in a library or is partially or completely automated. You should collect and store reusable process assets in some format, for example, a managed and controlled repository or a Process Asset Library (PAL).

As Figure 2-2 shows, there are several types of process assets. A standard process definition documents the architecture of a complete development process and the specified process steps that support that architecture. You can also document descriptions of product life-cycle models as part of the standard process definition or can develop and maintain them as a separate asset. The project process definition results when you select a product life-cycle description and use it as the basis for a project process architecture, which you then populate with specified process steps. You collect data that results when the project performs to the project process definition in the process measures database.



Source: Adapted from Paulk et al. (1993) and Kasunic et al. (1992)

Figure 2-2. Contents of the Process Asset Library

Sections 2.4.1.1 through 2.4.1.7 briefly discuss the types of reusable process assets that Figure 2-2 shows.

#### 2.4.1.1 Standard Process Architecture

The process architecture will facilitate process definition, construction, evolution, and reuse by identifying and defining standard process steps at a summary level. The process architecture identifies what partially ordered key process steps compose the overall process. The process architecture represents the process definition at the highest level of elaboration and has the following characteristics:

- Partially ordered sequence of process steps
- Key process steps that you define in terms of their functions, primary inputs, and major outputs
- Naming conventions and standards
- Standard process templates or formats
- Interface specifications
- Composition and tailoring rules

A process architecture does not decompose the key process steps in any great detail. You generally represent it graphically, such as through a data flow diagram, and support it by natural language text that defines the high-level scope, objectives, and function of each step.

### 2.4.1.2 Specified Process Steps

You use process steps to populate the process architecture. A process step is an activity or a primitive action that accomplishes a specific objective, such as to inspect a software requirements document. The difference between an activity and a primitive action is that an activity contains other steps, and a primitive action represents the lowest level of decomposition and is not further decomposed.

You generally specify a process step in terms of its supporting activities, methods, data elements, and work products:

- **Activities.** An activity is a step of a process that analyzes inputs and/or produces outputs to accomplish objectives that are derivative of the objectives of its containing process step. It describes what must be done without specifying how it should be done. Each activity can be assigned to an agent or an organizational position that has the responsibility and authority to control and complete the activity within the planned schedule and budget. An activity comprises other activities or unelaborated actions.

You can specify an activity informally or formally and can sequence it within the framework of its containing process step and, in some cases, within the overall process architecture. When specified and sequenced, an activity is enactable when you determine, define, and allocate the methods, practices, tools, and resources (dollars, time, equipment, personnel, etc.) to support the activity.

- **Methods.** Methods or specific guidance and criteria that prescribe a systematic, repeatable technique further support activities. Tools can support many methods. For example, several scheduling methods, such as Gantt and Program Evaluation and Review Technique (PERT) support the activity of developing a project schedule. A variety of automated tools, such as Microsoft Project, support both of these methods.
- **Data Elements and Work Products.** A data element is a piece or collection of information that you use as an input or output of an activity. Work products, such as a product development plan or a detailed design document physically represent activity inputs and outputs; that is, a work

product is any configuration-managed embodiment of one or more data elements. The output work product or part of the work product of one activity often serves as an input to one or more subsequent activities. Note that ability is the outcome of an activity in some cases; for example, knowledge or skill is generally the result of a training activity.

Work products are frequently subject to specific templates, such as the DOD-STD-2167A Data Item Descriptions. A degree of activity sequencing is implied when using data elements and/or work products as input to another activity. If the detailed design document is a necessary input to a coding activity, then you shall perform the activity that produces the detailed design document sometime before the coding activity, although not necessarily immediately before.

Sources that offer software development process steps and/or artifact specifications include DOD-STD-2167A, IEEE STD 1074, and ISO-9000. More comprehensive and specific guidance may be available in the industry. For example, comprehensive information system methodologies have been commercially available for several years. Some of these methodologies, such as the Navigator Systems Series<sup>®</sup>, are available in hypertext format with automated features that tailor the methodology based on specified drivers.

#### 2.4.1.3 Product Life-Cycle Model Descriptions

As shown in Figure 2-3, the product life-cycle perspective defines the primary states that a product reaches as it matures over its useful life. A given product life cycle will generally start when the product is conceived and end when the product is no longer available for use. For example, the life cycle of an aircraft encompasses the states it passes through from the time it was simply a statement of need until it is stripped for spare parts and retired from use.

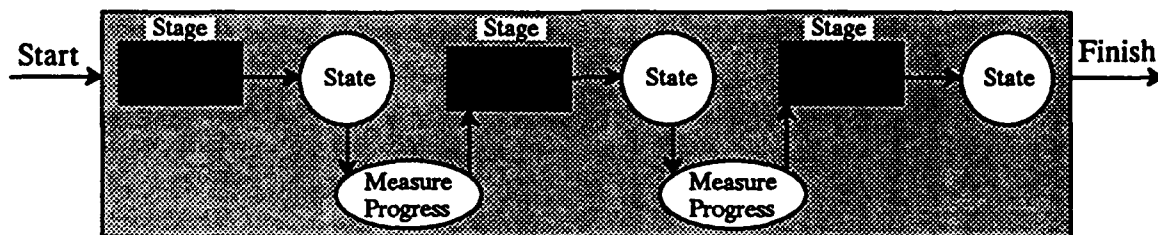


Figure 2-3. Product Life-Cycle Model

The life cycle simply brings the primary life-cycle states into view to allow measurement of progress in terms of how well a project is developing the product relative to the completion of each state. The life cycle does not offer any visibility to the life-cycle stages.

#### 2.4.1.4 Standard Process Definition

The organizational standard process definition documents the architecture of a complete development process that includes the process steps common across the organization. You should include guidelines and criteria for tailoring the process architecture and specified steps in this document. Additionally, you can include candidate life-cycle models in the standard process definition and map them to the process steps that will best achieve each of the life-cycle states.

The level of detail to which a standard process definition is documented varies from organization to organization. Process step scope; the cost, schedule, and staffing to enact a process step; and the



specific personnel, methods, practices, and tools used to perform each step are process drivers that generally vary from project to project. As a result, it is generally not effective to document a single standard process definition to an enactable level of detail.

The internal process guidebook appears to be the most common mechanism to date for documenting the standard process definition. Many organizations are developing an internal guidebook that often describes process in terms of policies and procedures. These internal guidebooks generally use a combination of graphical and textual notations and serve as the foundation for project process definitions.

#### 2.4.1.5 Project Process Architecture

The project process architecture describes the product life cycle you will use for a specific project. The project process architecture provides further visibility into the development process by identifying a set of partially ordered process steps within each of the life-cycle stages. Figure 2-4 shows the key process steps taken to reach each of the product life-cycle states.

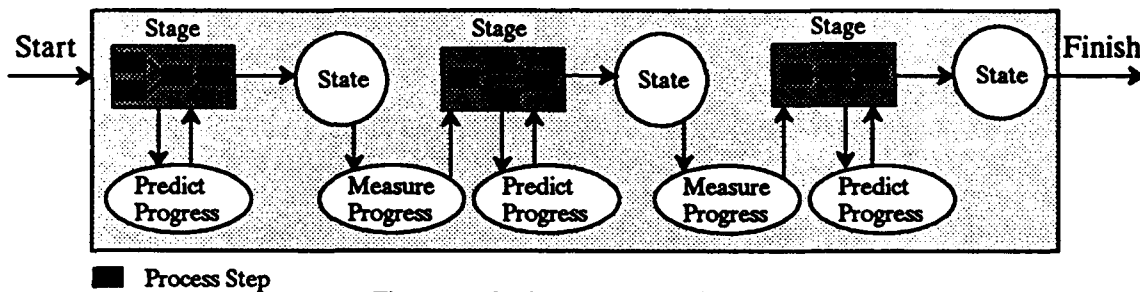


Figure 2-4. Project Process Architecture

Visibility into the process steps offers incremental<sup>1</sup> insights into product development and surfaces data that helps to predict progress toward the next state.

#### 2.4.1.6 Project Process Definition

A project process definition results when you select, tailor, and integrate a product life-cycle model, project process architecture, and supporting process steps into a coherent whole based on project-specific characteristics. In the project process definition, you tailor each of the process steps from the process step specifications that the standard process definition documents. If there is no existing specification for an identified step, then you specify and include it in the project process definition.

#### 2.4.1.7 Process Measures Database

The process measures database collects data that you generate as a result of enacting a project process and producing work products, as well as historical technical and management data regarding the use of process assets. Examples of process and work product data include estimates of software size, effort, and cost; actual data on software size, effort, and cost; productivity data; peer review coverage and efficiency; and the number and severity of defects found in the software code (Paulk et al. 1993).

### 2.4.2 PROCESS ENGINEERING PROCESS

In the past few years, SEPGs have followed the traditional process life cycle shown in Figure 2-1 to produce a single, comprehensive organizational process definition detailed to the enactable level that

reflects the various policies and procedures that the organization uses. Assuming that the organization contains multiple business areas or projects, a risk in taking this “all or nothing” approach is to produce a voluminous process definition that is unusable because it does not reflect business area-specific operational environments and other process drivers. You cannot easily adapt this comprehensive process based on the drivers of an individual project within a business area.

To mitigate this risk, the process engineer can develop the process definition as a set of process assets that reflect different levels of process elaboration and develop the definition based on process drivers. Figure 2-5 shows how to use the traditional process life-cycle phases to produce and continually improve standard process assets and how to use these assets engineer a process for an individual project. Figure 2-5 also illustrates the impact of process drivers that place requirements on each level of elaboration.

At the organizational level, process engineering starts by identifying existing process assets, which may include activity specifications, method descriptions, and other process information, material, and/or definitions, and by using these assets as the basis for creating an organizational process architecture. You may need to create from scratch parts of the process architecture that may be unique to the organization or where no material exists or is appropriate.

You may divide an organization into business areas or may characterize a coherent market by customers possessing similar needs. If this is the case, you can further detail one or more standard process definitions based on the unique drivers of the appropriate business areas.

The project process architecture addresses product-specific characteristics, such as the life-cycle model that a specific project uses, and is based on the standard process architecture that you developed for a specific business area or for the organization if business areas do not apply. The steps that the project process architecture identifies fully elaborate into the project process definition, which elaborates into an enactable process for the project.

Sections 2.4.2.1 through 2.4.2.6 discuss each of the process life-cycle phases in terms of the key process asset that results from the phases and the process drivers that place requirements on asset development.

#### **2.4.2.1 Architect Standard Process**

This phase involves defining organizational context, organizational process steps, and process standards. The process engineer analyzes and documents how to enact organizational procedures currently and resolves conflicting interpretations of these procedures to obtain a consistent view of how to perform standard operations. A process architecture synthesizes this information at an organizational level of elaboration, and depicts the major process steps within the organization and how those steps interact.

Standard process architectures will most likely vary from one organization to the next, based on unique process drivers. Process drivers that may place requirements on the identification and definition of process steps at an organizational level include:

- Investments (such as hardware, software environments, training, etc.)
- Organizational policies, procedures, practices, and standards
- Government rules and regulations (such as Federal Acquisition Regulations)
- Performance goals

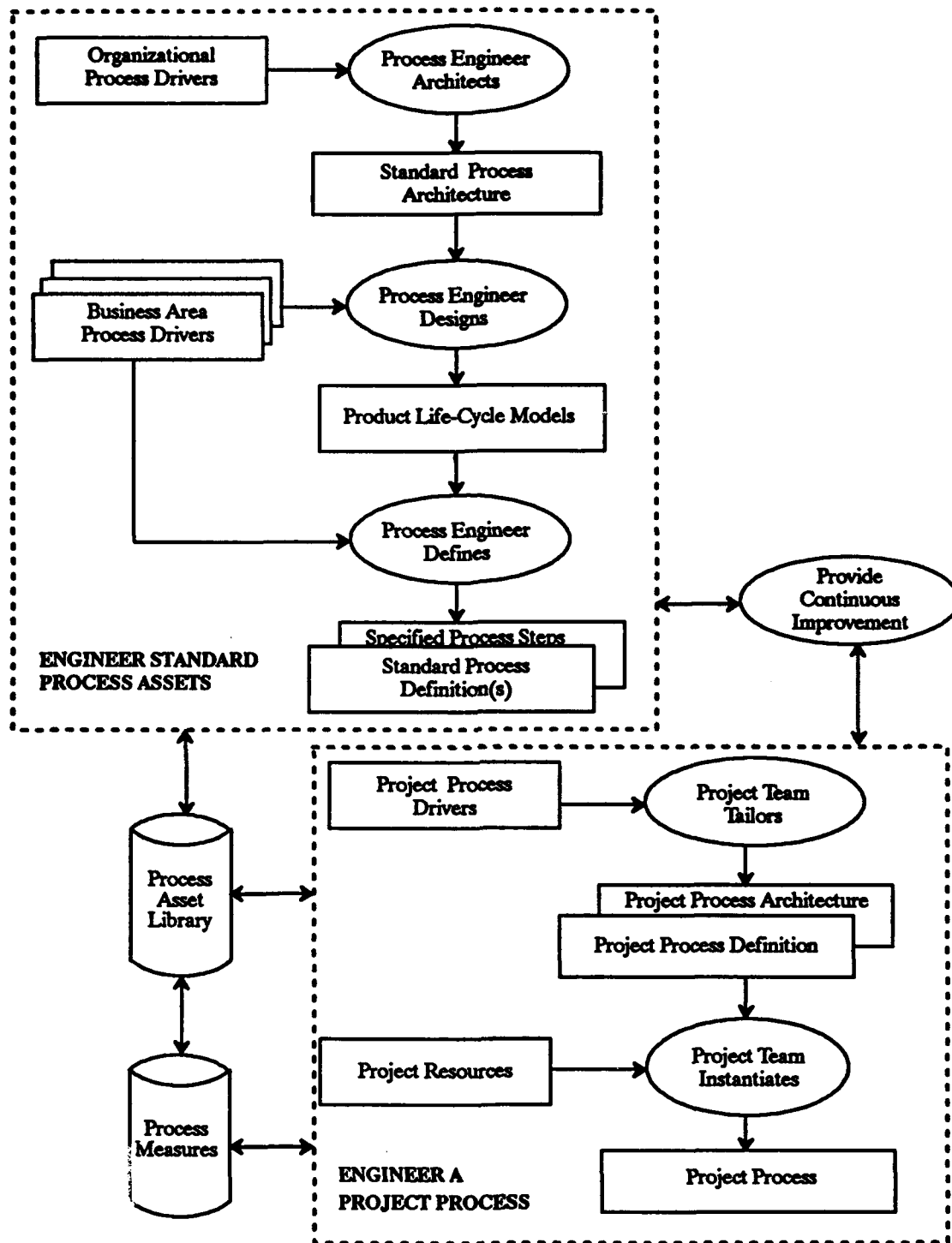


Figure 2-5. A Modified View of the Process Life Cycle

If an organization does not have further divided business areas, the process engineer should develop a single, standard process that includes the information that Sections 2.4.2.2 and 2.4.2.3 discuss.

#### **2.4.2.2 Design Standard Process**

This phase involves defining one or more product life-cycle models that the process will support and how the life-cycle models integrate with the conceptual framework that the architecture stage defines. By identifying business areas and their product lines, the process engineer can identify and describe the appropriate life-cycle models for key products in the product line and can begin to elaborate the life-cycle models with the partially ordered steps that will help achieve each of the key product's life-cycle states.

To describe product life-cycle models, the process engineer should first determine what constitutes an organization. An organization can be an entire company, a division, or a functional area within the division. One company may have several divisions, each with sufficiently different missions, customers, products, and cultures to warrant creating several business area-specific standard process definitions. Another company may determine that one standard process definition is sufficient because only one type of product is produced.

For example, the organization may produce products from several different domains or a coherent market characterized by customers possessing similar needs. If the organization operates in several different markets or it is a large entity, it will generally subdivide in a manner that makes the best business sense to that organization. However, in cases where the organizational boundaries do not reflect these different product domains, the process engineer should classify the typical products that are produced into one or more domains. If the resulting business areas are closely related, the process engineer may not consider the variability in the enactable processes within each area to be wide enough to warrant the development of separate standard process definitions. Instead, the different methods used to enact the standard process may accommodate subtle differences in the business operations.

Business area process drivers that may place requirements on product lines and the supporting life-cycle models include:

- Technology and business strategies
- Market trends
- Organizational infrastructure and culture

#### **2.4.2.3 Define Standard Process**

This phase involves decomposing and specifying the internal structure of each of the process steps identified in the standard process architecture based on business area process drivers. The standard process definition is the repository for the organization's standard process and specifies the key steps that the process architecture identifies into an operational, integrated, partially ordered series.

However, if an organization consists of business areas that have specific operational environments that are significantly different, the process engineers develop multiple standard process definitions based on each business area's unique process drivers, such as:

- Methods particular to the business area
- Risks specific to the business area

- Industry standards for the business area
- Existing product assets for the business area, such as development environments, reusable software, etc.

The process engineer should be able to define standard process definitions for specific business areas at a lower level of granularity than at the organizational level because of visibility into the product line, as Section 2.4.2.2 discusses.

### 2.4.2.4 Tailor Project Process

In this phase, the process engineer supports a project to tailor the project's process based on available standard process assets. First, a project selects the life-cycle model to use as the basis for the project process architecture. The project process architecture is a high-level description of a project's process to support the selected life-cycle model and guides the key product's development phase. It identifies the process steps that are performed, their connections to each other, and how they map to the product life-cycle phases. The process engineer assists a project to develop the project process architecture based on specific product drivers, such as:

- Product-specific requirements
- Product-specific development standards
- Product-specific methods
- Product-specific risks, such as performance or integration problems

They then elaborate the project process architecture into the project process definition by selecting process step specifications from the standard process definition and tailoring them to the project situation. Specifically, they develop the project process definition by selecting and tailoring the process step specifications that will meet the objectives of each product life-cycle stage, identifying the work products that will produce each step, and selecting the methods that each step or a set of steps uses. They tailor the project process definition based on project-specific drivers, such as:

- Selected product life cycle
- Project objectives, such as winning a follow-on contract or producing zero-defect software
- Project constraints, such as customer requirements or environment
- Project risks or potential problem areas, such as limited access to users

The process engineer and a project can define the project process definition either before the project team performs it or in parts, as long as they define each part before performance. In either case, the result should be a project process definition that represents the road map that the project team will follow to reach each life-cycle state.

This phase can be relatively straightforward if an organization subdivides into business areas and if each business area has defined its product line and the life-cycle models that best support each key

product in the product line. If this is the case, there may be several standard process definitions “on the rack” for a project to choose from and tailor to the project-specific drivers.

#### **2.4.2.5 Instantiate Project Process**

In this phase, the project manager instantiates the project process definition by using it as a template for the project schedule and by using the process step specifications as templates for specific project tasks, assigns specific agents to the each task, and estimates the time and effort needed to complete the tasks. Specifically, the project manager instantiates the project process definition by allocating or binding the steps documented in the project process definition with resources to produce an enactable process for the project team to follow. The project manager makes these allocations depending on such drivers as:

- Staff availability and years of experience
- Contract milestones dates
- Contract budgets

The result of instantiating the project process definition is a process ready for enactment. The process should document or provide a reference to everything that is needed to enact the process, including the resources necessary, the relationships of these resources to process steps, the products produced by these steps, and any constraints on enactment or resources. Resources include human process agents, computer resources, time, and budgets. Relationships refer to the estimation or assignment of resources to process steps to meet project objectives (Feiler and Humphrey 1992).

A project does not need to instantiate the project process definition completely at one time; as Section 2.4.2.4 discusses, it is often better to define and instantiate it in increments based on lessons learned and an increasing knowledge about project and product process drivers.

To put the instantiated process into practice, the project manager performs resource leveling to validate proper resource allocation to each task and initiates, monitors, and controls the execution of each task according to the project schedule. An enactable process consists of a process definition, required process inputs, assigned enactment agents and resources, an initial enactment state, an initiation agent, and continuation and termination capabilities. An enacting process may be in a suspended state if an assigned process agent is not available or other process constraints are not satisfied (Feiler and Humphrey 1992). The project team members enact the project process by following the schedule and performing each of their assigned tasks.

#### **2.4.2.6 Provide Continuous Improvement**

Process improvement stems from using and evaluating the baselined process at all levels of elaboration, from architecture to enactable process and from evaluating and monitoring the product being produced as a result of enacting the project process. Process evaluation techniques can include:

- Internal process assessments
- Ongoing measurements and metrics
- Post mortem evaluation

An internal process assessment assists an organization to determine and characterize its internal process maturity. The purpose of a process assessment is twofold. First, a process assessment provides an organization with a characterization of current process maturity and guidance on the key activities necessary for evolving to higher levels of process maturity. Second, a customer can use a measure of organizational process maturity as a way to assess the software development capability of potential contractors or vendors.

Measurements that a project manager might collect as an ongoing part of evaluating and monitoring product development include traditional cost, schedule, and size measures. In addition, a project could identify and classify errors discovered and solved or the type and severity of risks identified and mitigated when performing each activity of the process. The process engineer can also use cost, schedule, size, error, and risk measurements to build process metrics, such as an activity's average percentage of cost and time spent or the identification of the activities that typically discover or introduce errors and risks.

A post mortem evaluation of a project's process model and enactment might also identify potential reusable process assets, such as process(es), methods, tools, and measurements, as well as the rules for reusing these process assets. Process evaluation can include classifying and analyzing the different development process data by business areas to leverage similar process metrics of cost, time, errors, and risks.

### 2.5 SUMMARY

Many software organizations have expended a significant amount of time and resources to produce voluminous process descriptions. Often, these descriptions do not correspond to the processes actually performed during software development or maintenance. Some key problems may cause the disconnect between how a project actually performs to produce its required product and the organization's standard process model, including:

- Organizational process models developed at too high a level of detail
- Organizational process models defined at too low a level of detail
- Organizational process models that mandate inappropriate life-cycle model(s)

To a large extent, these key problems are symptoms of the difficulties currently encountered when attempting to tailor a standard process definition for a specific project. To resolve tailoring problems, it may help to consider the process model in terms of reusable assets, including:

- Standard process architecture
- Specified process steps
- Product life-cycle models

- Standard process definition
- Project process architecture
- Project process definition
- Process measures database

You can gain additional insight by understanding how to develop and elaborate process assets based on different levels of drivers, such as organizational, business area, product, and project. Section 3 shows how you can extend the ESP model to incorporate the steps for evolving a process through the different levels of elaboration and producing the supporting process assets.



*This page intentionally left blank.*

## **3. EXTENDING THE EVOLUTIONARY SPIRAL PROCESS MODEL TO PROCESS ENGINEERING**

The spiral cycle paradigm . . . can be extended to apply not only to the software product but also to the software process.

Barry Boehm and Frank Belz, *Experiences With the Spiral Model as a Process Model Generator*

### **3.1 OVERVIEW**

The ESP model is a framework for integrating both the management and development steps necessary for successfully producing a product. You can use the ESP model with any life cycle that describes the evolving maturity of a product. Because of this flexibility, the ESP model incorporates the steps for developing a process. Sections 3.2 and 3.3 present an overview of the concepts and principles of the ESP model and show how you can use the model to generate a process engineering model based on the issues and concepts that Section 2 discusses.

### **3.2 EVOLUTIONARY SPIRAL PROCESS MODEL OVERVIEW**

In product development, the desired objective is often to mature the product by a specified amount or to reach the next planned life-cycle state. You can use the ESP model to:

- Identify and/or adapt an estimated life-cycle model based on product and project objectives and constraints.
- Define a process model specific to your product and project situation by incorporating a life-cycle model into the conceptual ESP model.
- Expand the development process model to plan the process that the project will use to meet each of the life-cycle states or other major project milestones.

The concept of formally engineering a project-specific process model is relatively new to the industry; to assist the project manager with life-cycle decisions and process choices, many companies are investing in organizational process groups.

In several development environments, however, the product life-cycle and supporting process may be constrained at a certain level by the customer or by organizational policies and procedures. In this type of environment, the project manager can use the ESP model to ensure that such constraints are real and not perceived and to plan for a life cycle and supporting process in a manner that complies with actual constraints while taking advantage of any flexibilities that may exist. The project manager

should look for support from the organizational process group, if one exists, to perform this product planning activity.

After an estimated life cycle is established, a supporting process model can be developed by elaborating on the conceptual ESP model with the estimated life cycle. One of the key process management roles is to ensure that the product reaches the next defined life-cycle state by establishing and meeting objectives, goals, and success criteria. The project manager should plan the process for developing the product to the next level of life-cycle maturity and should manage to that process. The elaborated ESP model helps the manager perform this function by providing the framework for:

- Validating the next planned life-cycle state
- Identifying risks to and alternatives for reaching the next state
- Developing the plans and supporting process to reach the next state
- Monitoring the execution of the process
- Measuring progress against the plans
- Updating the plans and supporting process based on actual data and lessons learned

Another important management role is embedded in the Theory W (Boehm and Ross 1989) approach to project management: make everyone a winner. The theory postulates that one of the primary responsibilities of the manager is to make winners of everyone who is a stakeholder in the product and/or the process used to develop that product. The ESP model addresses issues in a way that makes everyone a winner by helping the project manager identify key stakeholders and by ensuring that the stakeholders remain involved in the appropriate level of project decision making.

The conceptual ESP model, shown in Figure 3-1, is described by five main steps and several specific supporting activities that focus on the management aspect of a complete development process. The ESP model is meant to be repeated using the knowledge gained and lessons learned from the previous cycle(s). The activities explicitly contained in the model are generic and can be used in conjunction with any life-cycle model. In addition, you can follow the ESP model activities to determine a life-cycle alternative that will adequately address your objectives and constraints, incorporate the life cycle into the model and project plans, and subsequently evolve the life cycle and supporting project plans as product development proceeds and objectives and constraints change.

You traverse all five steps of the ESP model one cycle at a time. A cycle is a complete traversal of all five steps that, when completed, matures the product by the amount defined in cycle-specific objectives and success criteria. A spiral is one or more cycles that, when combined, accomplish a specific objective, such as complete a project, product, work product, or other major milestone. A spiral may represent the complete life cycle or may only include the activities necessary to meet one or more of the life-cycle states.

Sections 3.2.1 through 3.2.5 offer a brief overview of each of the steps of the conceptual ESP model. Appendix A contains detailed activity specifications for each activity of the conceptual model.

#### **3.2.1 STEP 1: UNDERSTAND CONTEXT**

In Step 1 of the ESP model, understanding the context of the overall spiral as well as the current cycle establishes key ground rules for the product and the process used to develop that product. An Estimate

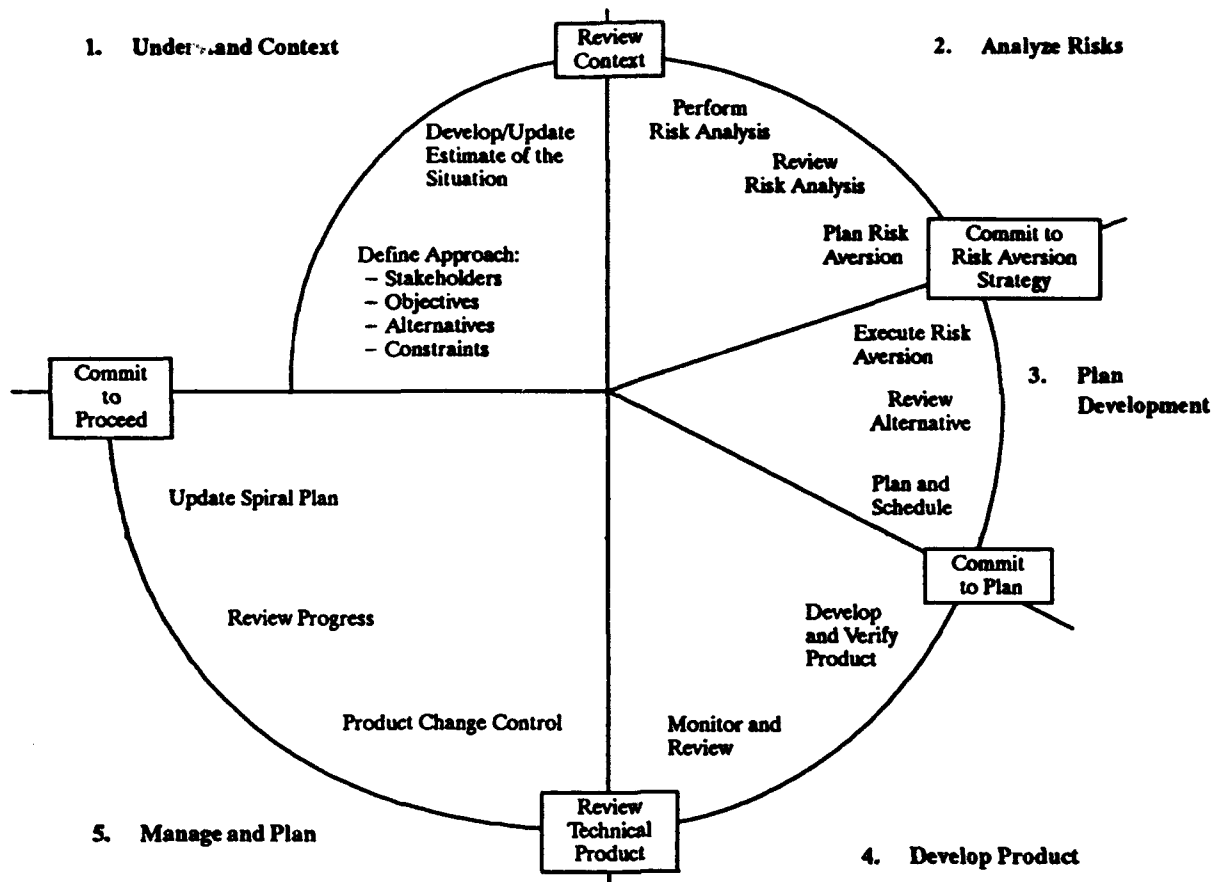


Figure 3-1. The Conceptual Evolutionary Spiral Process Model: A Management Process

of the Situation (EoS) is produced that documents stakeholders, objectives, alternatives, constraints, internal and external factors, and other information. The EoS is reviewed and updated each cycle. The cycle review at the end of Step 1 authenticates and establishes consensus on the general information that the EoS contains. The specific activities of Step 1 are:

- **Define Approach.** Understand the spiral and its supporting cycles in terms of stakeholders, objectives, alternatives, constraints, and other factors.
- **Develop/Update Estimate of the Situation.** To document the spiral and cycle context, create the EoS during the first cycles of a spiral and update it during subsequent cycles.
- **Review Context.** Ensure that all stakeholders review and approve of the appropriate parts of the EoS.

### 3.2.2 STEP 2: ANALYZE RISKS

During Step 2, you perform risk analysis to help identify, address, and eliminate risk items before they become threats to stakeholder win conditions, successful software operation, or overall spiral success (Boehm 1989). Risks are the potential unpredictable outcomes of alternatives and other factors. Risk analysis is a proactive management approach that focuses on what can go wrong and then attempts to keep it from occurring.

The specific activities of Step 2 are:

- **Perform Risk Analysis.** Identify, analyze, and evaluate risks.
- **Review Risk Analysis.** Team-review the risk identification, analysis, and evaluation activity results.
- **Plan Risk Aversion.** Select and plan for the most appropriate risk aversion strategy.
- **Commit to Risk Aversion Strategy.** Stakeholders commit to the selected risk aversion plan.

### 3.2.3 STEP 3: PLAN DEVELOPMENT

During Step 3 of the ESP model, you select one or more of the alternatives determined in the second step, but only after its risks have been averted to the extent possible. You plan and schedule the detailed development and development support activities from the selected development alternative(s).

The specific activities of Step 3 are:

- **Execute Risk Aversion.** Perform to risk aversion plan and select one or more development alternatives.
- **Review Alternative.** Review the selected development alternative(s).
- **Plan and Schedule.** Identify, organize, schedule, and assign resources to technical activities after any risks associated with development alternatives for the cycle have been averted.
- **Commit to Plan.** Review and commit to the development plan for the current cycle development activities.

### 3.2.4 STEP 4: DEVELOP PRODUCT

In Step 4 of the ESP model, you perform, monitor, and review the development and development support activities. The results of performing the activities should directly support the development goals and success criteria documented in the development plan.

The specific activities of Step 4 are:

- **Develop and Verify Product.** Perform the development activities to produce artifacts or advance product maturity, and verify that the artifacts or advanced product maturity meet requirements, development goals, and development success criteria. Generally, your long-range spiral plan is the specific product developed during the first cycle. A spiral plan is usually a set of planning documents, such as a risk management plan, product development plan, and project process definition.
- **Monitor and Review.** Monitor and review the technical development activities as they are performed.
- **Review Technical Product.** Review the product or part of the product developed to ensure that cycle objectives, development goals, and development success criteria were met.

### 3.2.5 STEP 5: MANAGE AND PLAN

During Step 5, you take stock of progress based on the outcome and lessons learned during the cycle, compare actual results against the cycle objectives, reevaluate and update spiral planning documents, and decide what to do next. The planning and management activities validate the work you performed in the cycle and adjust the project strategy based on the information that is newly available. The measurable development goals and success criteria defined in Step 3 and monitored during Step 4 are critical to this set of activities. They are used to judge whether the cycle is complete or needs to be iterated, or whether an alternative should be modified or abandoned.

The specific activities of Step 5 are:

- **Product Change Control.** Place the product or part of the product produced as a result of executing Step 4 development activities under product change control.
- **Review Progress.** Evaluate development plan actuals versus estimates, success criteria, and lessons learned. Update process drivers, including spiral objectives, success criteria, alternatives, constraint risks, estimates, and other information.
- **Update Spiral Plan.** Update all planning documents, as necessary, to record actual progress, reflect lessons learned, update estimates based on actual data, and update process drivers.
- **Commit to Proceed.** Review updates to the spiral plan and commit to proceed with the next cycle.

The five key steps and supporting activities of the ESP model are repeated using the knowledge gained and lessons learned from previous cycle(s) until the overall spiral objectives are successfully met or the spiral is suspended.

## 3.3 PROCESS ENGINEERING WITH THE EVOLUTIONARY SPIRAL PROCESS MODEL

Engineering the organizational-standard and project-specific processes requires continuous and systematic management of the many factors influencing the establishment and institutionalization of the process, such as people, technology, and change. The primary goals of process engineering with the ESP model are to:

- Develop and continually evolve standard, reusable process assets at the organizational level.
- Tailor the process according to project needs.
- Establish an effective interaction mechanism between the process group and the projects.

You can adapt the ESP model to represent the overall process engineering capability. Figure 3-2 shows the ESP-based process engineering model from both the organizational and project perspectives. As stated by Paulk et al. (1993), at the organizational level, you need to describe, manage, control, and improve the standard process in a formal way. At the project level, the emphasis is on the fit and usability of the project's defined process and the value it adds to the project.

Figure 3-2 shows how you should first attempt to understand your organizational context based on your current information level as the tactical plan from the Process Improvement efforts described in *Managing Process Improvement: A Guidebook for Implementing Change* (Software Productivity Consortium 1993a) documents. This document presents an agreed upon understanding of the current

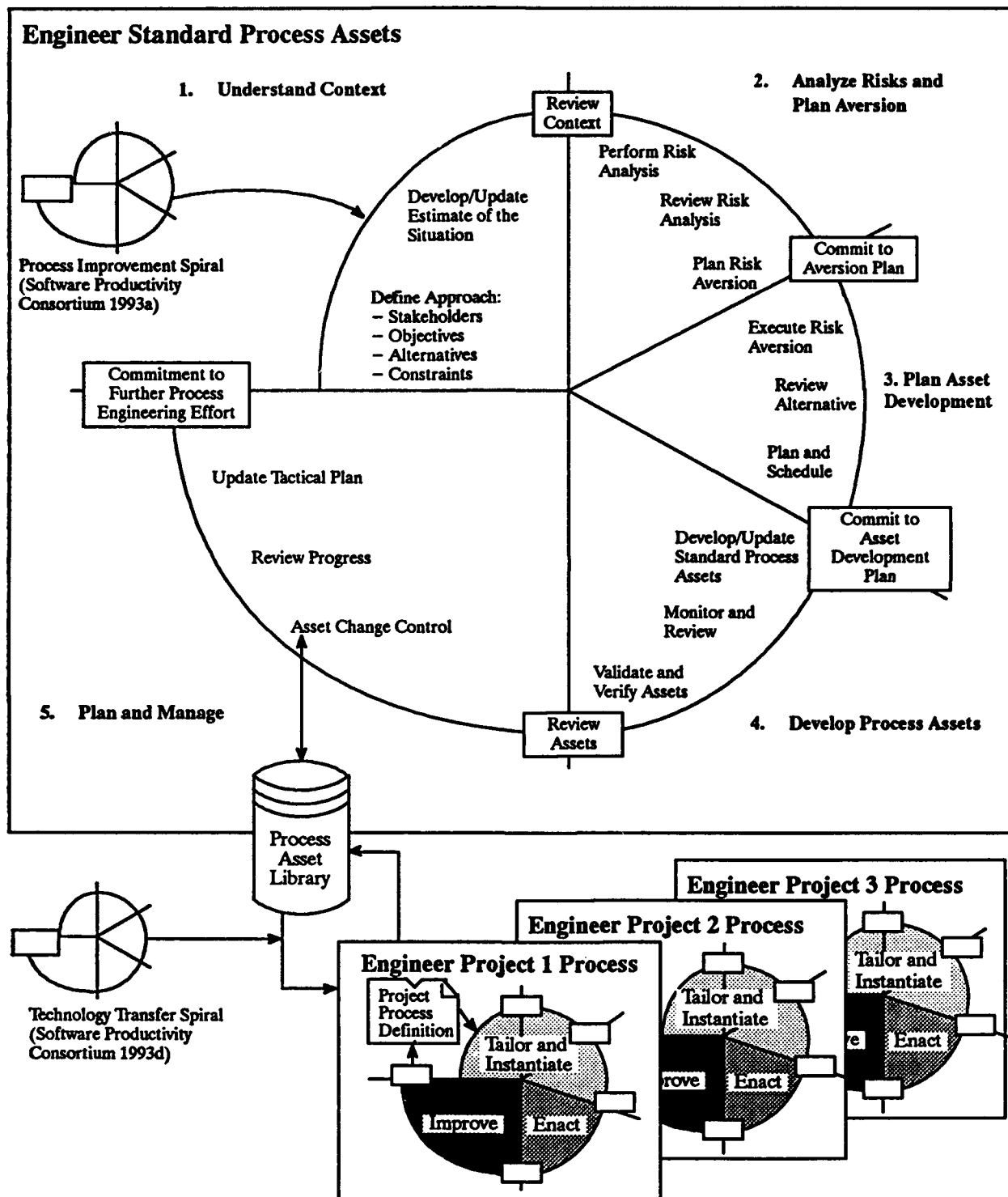


Figure 3-2. Process Engineering With the Evolutionary Spiral Process Model

process situation and defines improvement goals, alternatives, and priorities that you will need to progress to the next step.

You then analyze long-range goals and alternatives for achieving them for threats to success. This analysis will define areas of uncertainty or risks that may keep you from reaching your process improvement goals and help you to determine and plan risk aversion strategies that will help to minimize the impact of potential risks. Based on the understanding gained from executing your risk aversion strategy, you can justify the development of specific process assets and plan the next step in the evolution of the organizational process.

After you have reliable information on the process assets to be developed for this stage of evolution, you can implement them. This entails the more traditional engineering activities of design, implementation, and documentation. Verification and validation will be dependent on the correctness criteria for the assets.

You place approved process assets under change control and incorporate them into the PAL. They are now ready for use at the project level with the support of the technology transfer activity that *Using New Technologies: A Technology Transfer Guidebook* (Software Productivity Consortium 1993d) describes. This guidebook covers all the activities necessary to successfully transfer technology, including transition planning, training, and change control. Projects transfer and use the assets to tailor, instantiate, and enact specific process definitions based on unique process drivers. The project will return process usage data to the process database. Based on organization-wide process experience, you readjust your tactical plan and make a commitment to continue the process engineering effort.

### 3.4 SUMMARY

You can use the ESP model to improve an organization's or project's development process. Major features include a strong emphasis on risk management and the ability to respond quickly to problems or deviations from the development plan. These features, when used properly, can increase the productivity of the project team, improve the quality of its software products, and also enable the project to deliver on time and within budget more consistently.

You can use the ESP model, adapted from the original spiral model (Boehm 1986,1988), with any current development strategy or variation. Because of this flexibility, you can use the ESP model to produce many different types of products, including a standard process definition and other assets. Section 4 describes in more detail how an organizational process group can use an adapted ESP model to engineer standard process assets.



*This page intentionally left blank.*

## 4. ENGINEERING STANDARD PROCESS ASSETS

The as-documented, as-trained, and as-practiced process . . . is measured, maintained, and supported by the organization.

Mark D. Kasunic, et al.  
*Summary Report for the Process Definition Advisory Group*

### 4.1 OVERVIEW

Standard processes are those that a process engineer develops at an organizational level for use across projects. However, it is often difficult to develop standard processes that the project level can easily absorb or that a project can effectively tailor to address project-specific characteristics. This problem is characteristic of attempting to tailor the "navy blue, size 42 long" suit to fit every customer, from a basketball player to a jockey.

On the other hand, process development can be expensive. Because of the cost involved, there is considerable motivation for process commonality and sharing. This motivation is enhanced when different projects in the same organization tend to perform common activities.

### 4.2 ENGINEERING A STANDARD PROCESS

One of the ways to address tailoring issues is for the process engineer to engineer standard processes in terms of reusable assets. A project can use these reusable assets to construct the best possible development process based on unique objectives, constraints, risks, and other drivers. The process engineer evaluates, categorizes, and adds to the PAL process assets that the project constructed and enacted to build up a managed and controlled inventory of key "suits on the rack."

Process assets can take many forms and can be anything that is both useful and reusable in the engineering and enactment of processes on a project. Key standard process assets that the process engineer develops at the organizational level are:

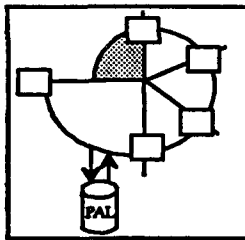
- **Standard Process Architecture.** The process architecture identifies and defines standard process steps at a summary level and rules governing relationships and composition.
- **Specified Process Steps.** Process steps populate the process architecture and are generally in terms of its supporting activities, methods, data elements, and work products.
- **Product Life-Cycle Models.** Product life-cycle models define the primary states that a product reaches as it matures over its useful life. A given product life cycle will generally start at the product's conception and end when the product is no longer available for use.

- **Standard Process Definition.** The standard process definition documents the architecture of a complete development process containing the process steps that are common across the organization. Additionally, the process engineer can include candidate life-cycle models in the standard process definition and map them to the process steps that will best achieve each of the life-cycle states.
- **Process Measures Database.** The process measures database is a repository for data collected as a result of enacting a project process and producing work products.

An organization often subdivides into business areas or produces different types of products. If this is the case, the process engineer can tailor and map the standard process architecture and specified process steps to product-line-specific life-cycle models to produce a standard process definition that is specific to each business area.

You can use the ESP model described briefly in Section 3.2 with any current development strategy or variation. Because of this flexibility, you can use the ESP model to produce many different types of products, including standard process assets. Figure 4-1 shows how the process engineer can adapt the ESP model slightly to help describe, manage, control, and improve standard process assets. Sections 4.2.1 through 4.2.5 elaborate each of the five major steps of the standard process engineering spiral, as Figure 4-1 shows, by describing the type of activities the process engineer performs for each step.

#### 4.2.1 STEP 1: UNDERSTAND CONTEXT



In this step, the process engineer should attempt to understand the overall context for the entire standard process engineering spiral, as well as the context for the current cycle, given the current level of information and knowledge.

The process engineer should define the organizational process engineering spiral and its supporting cycles in terms of stakeholders, objectives, alternatives, and constraints. A primary input into this activity is the tactical plan for the SEPG, a document that the process improvement spiral produces and that *Managing Process Improvement: A Guidebook for Implementing Change* (Software Productivity Consortium 1993a) elaborates on. The tactical plan that the process improvement spiral produces is equivalent to the organizational process engineering spiral plan, that is, the long-range plan that guides the objectives and the activities to meet those objectives for developing the organization's standard process. The process engineer updates the tactical plan during each cycle to reflect the near-term or specific cycle objectives and the activities to meet those objectives.

Specifically, the process engineer defines the approach for the organizational process engineering spiral by:

- Identifying and involving key persons having a stake in the process definition effort, including selected project managers and practitioners. Note that the stakeholders may be different for each cycle of the spiral.

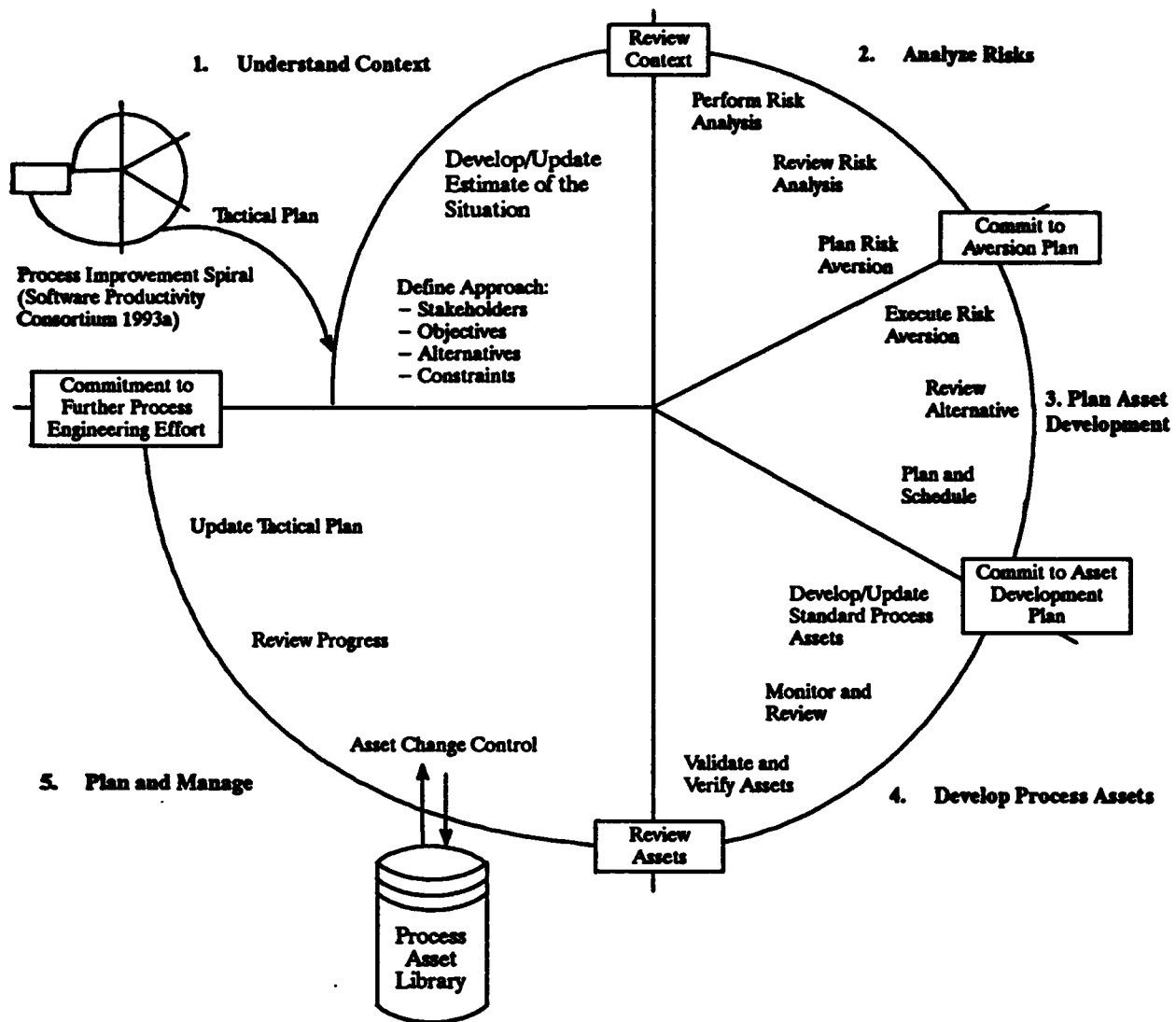


Figure 4-1. Standard Process Engineering Process

- Determining the expectations and objectives. Generally, the primary objective of Cycle 1 of the spiral is to establish and baseline the organization's "as-is" processes or existing process capability. During subsequent cycles, the process engineer updates these objectives to reflect the "to-be" process objectives, that is, objectives for moving the process toward a new process state one cycle at a time.
- Determining or verifying the alternative(s) to guide the overall effort, such as the SEI Capability Maturity Model (CMM). This typically occurs in Cycle 1. During subsequent cycles, the process engineer updates the alternative to reflect lessons learned.
- Identify alternatives specific to the objectives of the current cycle. For example, if the objective of the cycle is to begin to develop project management process assets, the process engineer should identify what assets to consider developing during the current cycle, what reuse alternatives may exist, the possible levels of abstraction to which the assets can be decomposed and defined to the enactable level, and process notation alternatives.

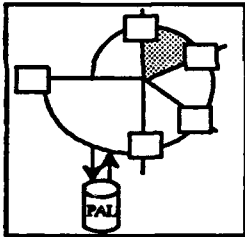
- Documenting any constraints the process engineer has while pursuing both the spiral and cycle objectives.

The process engineer should document the information learned during the Define Approach activity using a method such as the EoS. The process engineer completes an EoS during Cycle 1 of a spiral and updates it during subsequent cycles. The EoS can be a standalone document or can be incorporated into the tactical plan. At a minimum, the EoS should identify, analyze, and document the scope of the organizational process engineering spiral and its process drivers.

Process drivers are key characteristics that directly affect the development of the standard process. The process engineer can identify process drivers from sources such as the process engineer tactical plan; reports resulting from the organizational process assessment; client and business area policies, procedures, and standards; and interviews with key project personnel at all levels.

After collecting sufficient information on the organization's current process maturity and intended vision at the overall spiral and current cycle levels, the process engineer should ensure that all stakeholders understand and have the same interpretation of the situation, especially the objectives and alternatives for meeting those objectives. Stakeholders should review the EoS document, as well as reexamine the tactical plan. This activity should set stakeholder expectations and establish buy-in and commitment.

#### 4.2.2 STEP 2: ANALYZE RISK



This step analyzes the threats from moving from the current process maturity state, or the "as-is" process, to the new process state, or the "to-be" process, documented in the form of process assets.

The process engineer should first analyze the proposed changes to the people, process, and technology that the tactical plan and the EoS documents to:

- Identify the factors that threaten success to the objectives to both overall process improvement and current cycle process asset development.
- Analyze risk areas for the likelihood of occurrence and impact should they occur.

After the process engineer has identified and analyzed how the process will fail, the next activity is to determine the most appropriate strategies for reducing the likelihood or impact of failure. The output of these activities is the draft risk management plan (RMP). Like the EoS, the RMP can be a standalone document or can be incorporated into the tactical plan. The process engineer drafts the RMP in Cycle 1 and updates it during subsequent cycles.

The process engineer should review the draft RMP before taking any action. Because risk is subjective and often things are overlooked or given too much importance, it is important that all stakeholders reach consensus on the analysis and understand the risk situation before continuing. The process engineer revises the RMP to reflect the group's understanding. Review comments may identify the need to repeat some or all of the previous risk activities.

When the process engineer understands the risks to the organizational standard process objectives as well as possible given the current level of knowledge and information, determine how to expend resources to avert risks and reduce the chances of failure while making some substantial improvement to the organization's process maturity. The process engineer should consider risk reduction and asset generation objectives and strategies in an integrated fashion to trade off the cost of acquiring more certainty on the process definition approach with the need to transfer process capability to end users in a timely manner.

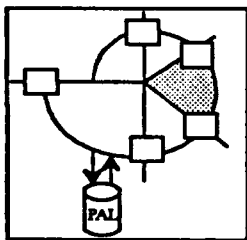
The process engineer may wish to plan for separate risk aversion efforts when the approach to reducing the risk will require an iterative approach. These efforts are risk-reduction subspirals. The process engineer should treat them as subprojects, chartered to accomplish a specific risk reduction objective and empowered sufficiently to reach the objective without impact from the main-stream development. Interfacing constraints will also pass as key inputs to the risk-reduction subspiral to ensure that the results integrate with any asset development that may be occurring in parallel within the main spiral.

Ideally, the entire SEPG should evaluate and agree to the risk aversion strategies to be followed. The process engineer should:

- Document the selected risk aversion strategies in the draft RMP
- Estimate, evaluate, and include cost and schedule for each risk aversion strategy in the draft RMP
- Make sure that aversion strategies are planned so that their results are available in time to plan for the asset development activities that depend on them

After completing the risk aversion plan for the current cycle, the process engineer should reach consensus with the appropriate stakeholders on what is to be done and how it is to be performed. Specifically, those responsible for executing the plan should support it.

#### 4.2.3 STEP 3: PLAN ASSET DEVELOPMENT



In Step 3, the process engineer plans and schedules the asset development activities to be performed in Step 4. The planned activities should be driven from the process asset development alternatives agreed to for the cycle.

The process engineer executes risk aversion activity after planning and approving it in Step 2. The outputs and products of the risk aversion activity furnish information to guide the asset generation activities. Examples of risk aversion activities may consist of prototyping efforts for process definitions, examining the use of procedure to direct organizational process interfaces, or determining and documenting best practices from many projects.

During Cycle 1, the process engineer should execute risk aversion strategies that justify the alternative(s) selected to guide the overall process improvement effort, such as the SEI CMM. During subsequent cycles, the alternative should be updated to reflect lessons learned.

The process engineer also should execute risk aversion strategies to verify the alternatives specific to the objectives of the current cycle and should attempt to avert the risk of defining process at an

in appropriate level by analyzing which process steps can standardize at each level of abstraction. This involves deciding at what level of abstraction certain process steps can be decomposed and defined to the enactable level. For instance, if Verify Documentation is an activity that a project will enact using the same inspection process independent of the business area, then this activity decomposes down to the enactable level. But a project may perform the Design Software System activity using different development paradigms in different business areas; therefore, this activity will remain a black box at the organizational level and will be further elaborated in the business area standard process definitions.

As Section 4.2.2 discusses, complicated or longer term risk aversion activities may spin off into their own spirals and be managed independently from the main-line development that Step 3 discusses. The results of the risk aversion activities will generally justify a particular process asset development alternative, including what assets the process engineer will generate during the current cycle and to what level of definition abstraction.

The process engineer should review and present the results of the risk aversion activities to the appropriate stakeholders to reaffirm the overall alternative for guiding the process improvement effort and to decide on the specific process asset development alternative for the current cycle. After gaining commitment from the stakeholders, the process engineer can place the RMP under configuration control and begin to develop detailed plans for developing process assets.

The process engineer uses the information generated by the risk aversion activities to plan and schedule the asset development that can be achieved during the current cycle. By using the results of the risk aversion activities, the process engineer should be able to document asset development plans that have a high probability of producing the desired products within the allowed cost and schedule.

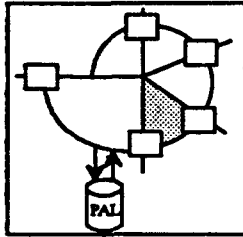
The main output of the plan and schedule activity is a development plan that plans for the process assets to be developed in the current cycle. Specifically, the plan should:

- Establish process asset development goals and associated success criteria that support the current cycle objectives.
- Estimate the scope of the assets that the process engineer will develop in the current cycle.
- Estimate asset development cost and schedule, and allocate resources.
- Identify the activities or methods the process engineer will perform in the current cycle, such as reuse adoption activities for establishing the reusable process database as described in Software Productivity Consortium (1993b) or the Integrated Computer-Aided Manufacturing Definition (IDEF) method (SofTech 1981) for representing process steps.
- Define/redefine activities, methods, and resulting artifacts if necessary.
- Sequence the activities if the selected method does not sequence them.
- Select and allocate supporting tools, such as ADW, an automated tool supporting the IDEF method (SofTech 1981).
- If appropriate, define work packages or the lowest work breakdown structure (WBS) level for the key asset development activities that the current cycle defines.

The process engineer should document the detailed asset development plan for the current cycle as an evolution of the tactical plan.

The appropriate stakeholders should have the opportunity to review and comment on the results of the process asset development planning and scheduling activities. The process engineer may submit the draft development plan to the appropriate stakeholders for review and update the development plan in response to comments.

#### 4.2.4 STEP 4: DEVELOP PROCESS ASSETS



This step is concerned with developing process assets that a project can employ, including tailoring guidance and training as necessary to support institutionalization on individual projects.

During this step, the process engineer performs to the asset development plan developed and committed to in Step 3 to produce process assets or to advance asset maturity. Specifically, the process engineer should perform the following activities as part of this step:

- **Develop/Update Standard Process Assets.** During this step, the process engineer produces the process asset(s) or part of the process asset(s) necessary to meet the cycle objectives and the development goals and success criteria that previous steps established.
- **Monitor and Review.** Continually monitoring asset development helps to maintain control over associated costs, schedules, and risks. There are a variety of standard project management methods and tools to use during this activity. Traditional scheduling tools help track progress and determine how actual costs and schedules compare to estimates. A risk referent measures acceptable risk for individual and overall risks. As the process engineer performs asset development activities, the risks associated with the activities are monitored and tracked against that referent.

Reviews facilitate SEPG interchange that can affect the direction of the assets that the process engineer developed during the current cycle. Periodic reviews demonstrate to the SEPG and other stakeholders that the asset development activities are proceeding according to plan and that the resulting artifacts are meeting the cycle objectives and process asset development goals. Reviews are also a way to obtain a commitment from stakeholders to proceed with the asset development activities for the cycle as documented in the tactical plan. At these reviews, the SEPG can decide to reallocate resources, replan schedules, or reassess risks for the current cycle development activities.

- **Validate and Verify Assets.** During this step, the process engineer validates the process asset(s) or part of the process asset(s) against cycle objectives, development goals, and success criteria. When sufficiently documented, the process engineer should verify process assets for correctness and usability. The process engineer may wish to use walkthroughs, inspections, or simulations as possible verification methods. For critical components, the process engineer may need



to execute pilot projects, perhaps as subspirals. The process engineer may have to iterate the asset generation and training with the information learned from the validation and verification activities.

- **Review Assets.** The SEPG and other stakeholders should participate in a final technical review of the process assets and agree that the assets satisfy the requirements they were intended to serve. This review is a final opportunity for the stakeholders to ensure that they met cycle objectives, as well as process asset development goals and supporting success criteria. Ideally, this review should be a formality because the ongoing monitor and review activity should have identified and corrected most problems.

Generally, the first process asset that the process engineer develops is a detailed baseline of the existing process capability. This activity usually occurs in Cycle 1 and involves evaluating the existing process improvement infrastructure and capability to establish a common understanding and baseline of the current organizational process.

After baselining the existing capability, the process engineer conducts activities in subsequent cycles to improve process in a planned and controlled manner. The process engineer evaluates and documents alternatives to ensure that assets are at the appropriate level of abstraction, in time to support increasing process maturity needs, and useful to individual projects within the organization.

Sections 4.2.4.1 through 4.2.4.6 discuss the process capability baseline that Cycle 1 establishes, as well as the organizational process assets that subsequent cycles develop.

#### **4.2.4.1 Process Capability Baseline**

After an organizational process maturity assessment, process engineers often attempt to define the ideal way to conduct business instead of first analyzing what is currently done in the organization and why. This phenomenon often occurs because organizations feel that documenting current, ad hoc practices is a waste of time and resources. After an assessment, management and practitioner often assume that what is wrong with current processes is well known and that payoff from improvement efforts will result only when they define and institutionalize the right processes. The entire organization is eager to define the “right” processes and realize process improvement benefits as soon as possible.

However, to improve process in a timely and controlled manner, the first asset that the process engineer produces should be a baseline of existing process capability. Analyzing existing processes is an essential first step for introducing effective change by identifying high-impact process improvement areas, leveraging from effective and familiar practices, and determining incremental improvements that are easier to introduce and absorb into the organization. Building new processes on top of a current process baseline, rather than installing an entirely new process, is important to keep the organization from becoming disoriented during the improvement program (Curtis, Kellner, and Over 1992).

Analyzing and documenting how organizational procedures are enacted currently and resolving conflicting interpretations of these procedures helps to establish a consistent view of how an organization performs standard operations. The process engineer can further define and propagate where the current practices are effective and can examine current practices that are not as effective to establish an understanding of process requirements. Another important view into the organization to gain at this time is the existing process structure, that is, the different business areas and their relationships to each other and to the organization.

The process engineer need not document the process capability baseline to an enactable level of detail; rather, it should document the existing key steps that compose the current process and should specify these high-level process steps in terms of brief description and entrance and exit criteria. The architecture will help to identify where there are gaps and redundancies in current practices, and the specifications will show where activity inputs and outputs are not being used effectively, are repetitive, or are simply unnecessary.

As part of establishing the process capability baseline, the process engineer should note existing process steps that are generally effective or could become effective with some adjustment. The process engineer can update these existing assets in subsequent cycles and define them to an enactable level of detail.

#### **4.2.4.2 Standard Process Architecture**

Another asset developed at the organizational level is the standard process architecture. The architecture represents the process at the highest level of abstraction; it identifies what key steps compose the overall process and a partial ordering of those steps. Specifically, the process engineer defines the standard process architecture by:

- Identifying and analyzing organizational process drivers, including an understanding of the different business areas within the organization if applicable
- Developing a common process vocabulary and process asset development standards
- Determining the key process steps that are generally applicable across all business areas
- Describing the key process steps at a high level in terms of their functions, primary inputs, and major outputs
- Achieving consensus on the standard process architecture by all stakeholders
- Baselining the process architecture and including it as part of the PAL
- Training all process engineers and other stakeholders within the organization on the organizational process architecture
- Monitoring the use of and continually improving the standard process architecture

The process engineer generally represents a process architecture graphically, such as through a data flow diagram, and supports it by natural language text that defines the high-level scope, objectives, and function of each step.

#### **4.2.4.3 Specified Process Steps**

The process engineer further decomposes, specifies, and documents each of the key process steps of the architecture into relatively self-sufficient entities by providing textual detail, including selection, tailoring, instantiation, and training guidance. There may be more than one specification that can apply to a certain step or series of steps in the process architecture, especially if the organization comprises multiple business areas. For example, certain testing process steps may contain numerous

activities for a business area that develops air traffic control software that are not applicable or necessary for a business area that develops financial information systems and vice versa.

Generally, fully specified process steps are a combination of graphical representations and formal textual descriptions. The process engineer specifies each process step by:

- Developing standard process step composition and interface rules, including specification templates or formats
- Decomposing each process step into supporting activities and unelaborated actions
- Formally specifying each included process step by verifying and elaborating the high-level descriptions developed with the process architecture and defining appropriate entrance, exit criteria, and verification criteria
- Identifying and specifying the work products that are produced as a result of performing the step
- Identifying any measures that should be taken when performing the step
- Describing candidate methods, practices, and tools that can be used to perform each step
- Describing the skills an enactor should possess to successfully perform the step
- Providing cost and schedule estimating heuristics for each specified step
- Developing tailoring and instantiation guidance for each specified step
- Achieving consensus on each specified process step by all stakeholders
- Baselineing each specified process step and including it as part of the PAL
- Monitoring the use of and continually improving the specified process steps

The process engineer can use notations to specify process steps in a consistent way. In general, notations provide the format of a process step description, including inputs, outputs, and start and completion criteria. A notation can be an informal description of an activity using natural language and standard formats, or it can be a formal specification language. Informal notations are sufficient for practitioners' process manuals; formal notations lend themselves to process automation or the use of a machine to automate all or part of the software development process.

When developing an internal process guidebook, an organization can use the *Process Definition and Modeling Guidebook* (Software Productivity Consortium 1992a). This guidebook discusses several common notations and provides a tailorable approach for process definition and modeling. Specifically, the guidebook provides a flexible set of templates and techniques for capturing and representing process in terms of the relationships and constraints between and within activities, artifacts, and supporting resources. The techniques that the guidebook documents attempt to represent the process in terms of an optimal combination of text and graphics.

Not all of the information in the paragraphs above needs to be specified at one time. For example, the process engineer could document cost and schedule heuristics after using the process steps and

collecting data. Additionally, it may not be effective to fully specify process steps at the organizational level if there is significant variability in the way projects perform the steps within the business areas. At a minimum, the process step specifications should include a description, inputs, outputs, entrance criteria, and exit criteria.

Appendix B to this guidebook provides a useful “starter set” of software development process step specifications to adopt verbatim or adapt to reflect the business area’s existing infrastructure, culture, language, policies, procedures, standards, and other process drivers.

#### **4.2.4.4 Product Life-Cycle Models**

An organization often divides into multiple business area, especially when producing products for a variety of contractual and/or commercial customers and users. Because these products have unique features and characteristics, a single product life cycle may not be appropriate for all situations. Therefore, the organization may identify more than one product life cycle that the projects can use.

The process engineer defines different product life-cycle models by:

- Reviewing the existing process structure or different business areas and their relationships to each other and to the organization
- Determining the product line or collection of existing and potential products for each business area
- Defining life-cycle models for key products in each product line
- Achieving consensus on the product life-cycle models by appropriate stakeholders within each business area
- Baselining the product life-cycle models and including them as part of the PAL
- Monitoring the use of and continually improving the life-cycle models

The process engineer typically can obtain these product life cycles from software engineering and literature and modify them for the organization or business area (Paulk et al. 1993).

#### **4.2.4.5 Standard Process Definition**

The standard process definition is a view into the process that results when the process engineer integrates the standard process architecture, process steps, and product life-cycle models into a coherent whole. Depending on the organization, it may be effective to develop multiple standard process definitions that support different business areas or to show how the standard process can support different product life cycles.

Specifically, the process engineer develops the standard process definition(s) by:

- Identifying and analyzing business area process drivers
- Tailoring the organizational process architecture to be more supportive of specific product life-cycle models

- Populating the tailored process architecture with key process steps to achieve each of the life-cycle states
- Specifying or tailoring the key process steps based on business area process drivers
- Developing general guidance on how to tailor the standard process definition(s) for each project
- Achieving consensus on the standard process definition(s) by appropriate stakeholders within the business area
- Baselining the standard process definition and including it as part of the PAL
- Monitoring and continually improving the standard process definition
- Training process engineers and project managers on the standard process definition(s) and how to tailor them for each project within the business area

After the process engineer elaborates the standard process definition(s) to a standard process definition level of abstraction, they are ready for any of the projects within the business area to use.

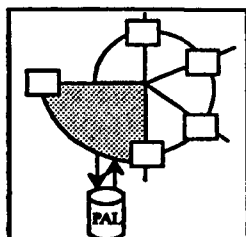
#### 4.2.4.6 Process Measures Database

The process measures database is a repository for data collected as a result of enacting a project process and producing work products. Specifically, the process engineer establishes the process measures database by:

- Determining the type of repository to be developed for process measures, that is, automated, manual, or a combination of the two
- Establishing the repository and the rules and standards for populating it
- Collecting measures at the project level, such as estimated and actual size, effort, and cost data; peer review coverage and efficiency; and number and severity of product defects found

The process engineer can use cost, schedule, size, error, and risk measurements collected at the project level to build process metrics, such as an activity's average percentage of cost and time spent or the identification of the activities that typically discover or introduce errors and risks. The process engineer should define key metrics to be developed based on the collected measures to monitor process use, provide process step cost and duration-estimating heuristics, and identify process improvement areas.

#### 4.2.5 STEP 5: PLANNING AND MANAGEMENT



The planning and management step controls the process assets and updates the overall tactical plan for the organizational process engineering spiral.

After the process engineer meets development goals and success criteria and after the SEPG and other stakeholders approve the resulting artifacts that Step 4 produced, they place organizational

process assets under change control. This is necessary because assets will likely evolve over time, and the process engineer must be able to trace project use to one of many versions. After assets are placed under change control, they are added to the PAL and made available for the entire organization.

At the appropriate time, the process engineer coordinates the transition of the process assets with every project that will use them. The process engineer works with the project manager and other team members to develop specialized tailorings of the process assets.

The process engineer may need to provide additional process support to the projects if they are having difficulties using the process assets. This support is essential to successful technology transfer. When projects have process difficulties, the process engineer should make available the appropriate assistance. For more information on how to transfer process assets and other types of technology, refer to *Using New Technologies: A Technology Transfer Guidebook* (Software Productivity Consortium 1993d).

The process engineer captures and uses information on project use of the organizational standard process, including specialized tailorings, lessons learned, and measurements, to assess root causes of process defects and to determine improvement alternatives.

At the end of each cycle of the organizational process engineering spiral, the process engineer updates the tactical plan, as necessary, to reflect lessons learned, estimates based on actual data, and updated process drivers. The tactical plan becomes a living document and serves as the repository of spiral process history and change. At this time, plan for the first three steps of the upcoming cycles. Plan the next cycle planning and risk aversion activities to estimate the situation, analyze and avert risks, and plan asset development.

At the completion of each cycle or stage of process capability evolution, the process engineer and other appropriate stakeholders should renew their commitment to continue with the process engineering efforts. The stage may take a long time to complete, new stakeholders may join the team, others may leave or be replaced; therefore, the process engineer should reinsure sponsorship and buy-in to the next cycle of process engineering activity.

### **4.3 OPTIMIZING ORGANIZATIONAL PROCESS ASSETS**

Engineering a process that meets a project's needs is a critical part of producing high-quality software within budget and on schedule; it can also be time consuming and costly. Level 3 of the CMM requires guidelines and criteria for tailoring the organization's standard process for use on projects. The process engineer should develop guidance on why, when, and how to tailor the standard process definition for use at the project level and maintain this guidance as a process asset. However, even with guidance, identifying project process drivers and then manually tailoring and instantiating the project process can be a resource- and time-intensive task. In addition, tailoring guidance and the resulting procedures often require that the purpose and rationale for any tailoring of the existing standard process definition be documented, justified, and approved.

As organizations reach the highest levels of process maturity, it may be possible to optimize traditional process engineering activities by developing enhanced tailoring and instantiation guidance based on business areas and product lines. The process engineer can develop specific tailoring and instantiation guidance based on fundamental project process drivers as a solution for minimizing the time and resources that projects must spend to define their project-unique process adequately. By providing

tailoring guidance based on inputs such as the estimated project size, cost, schedule, risks, and available resources, the process engineer can help a project define its process by providing answers to predefined questions, such as:

- What is the right set of process activities and the appropriate process model for the project?
- How should the activity specifications be tailored?
- What should the artifacts be?
- How should resources be allocated to activities?
- What methods, tools, and techniques should be used, and how should their use be customized?

As the project learns more about its process drivers in subsequent cycles, further guidance is necessary to assist with process evolution activities. The project may require guidance on how to evolve the project process definition given the current level of understanding and knowledge about the process drivers, such as actual product measurements, including cycle costs, schedules, risks, resources, and assets.

Providing tailoring and instantiation guidance based on specific variations and priorities of process drivers would appear to be a monumental task. First, the standard process definition and supporting environment should encompass all basic life-cycle, management, and support activities. Second, much knowledge and experience should accumulate as a result of using the standard process definition to determine the best response(s) to each process driver. Third, it is likely to take significant time and effort to define standard process drivers and to derive appropriate conditions, criteria, and solutions to instantiate mechanically the standard process definition to produce a project-specific definition.

The STARS and SEI are currently examining these difficulties. Specifically, the STARS/SEI team has proposed to define and document process assets or the components for constructing project-specific processes. The initial definition and documentation effort will include the collection, cataloging, analysis, partitioning, distillation, and synthesis of software processes by that industry, the government, and academic organizations have submitted. The resulting process assets, including adaptation, tailoring, installation, and evolution guidance, will be piloted and tested (Over 1991, 45-60). The SEI has released an initial release of these process assets (Software Engineering Institute 1992).

Figure 4-2 shows the Synthesis process model, which you could also use to address the problem of process creation, tailoring, and instantiation. The Software Productivity Consortium (1993c) developed the Synthesis process model to support better use of expertise and knowledge about a set of similar problems and associated solutions within a specific business area or group of projects supporting a similar product line or serving a similar customer base. Synthesis defines and integrates two processes: Domain Engineering and Application Engineering.

The Synthesis process model extends the concept of defining the process model to the business area level of abstraction by providing extended domain engineering guidance. Where a business area refers to a coherent market that potential customers possessing similar needs characterize, a domain is the complete product family and associated production process supporting a product line. Domains can cut across the industry to the extent that different companies in the industry have similar business objectives, produce similar products, or serve a similar customer base. For example, each of several large companies may have domains that build similar aerospace systems or subsystems, such as aircraft avionics, missile controls, or helicopter flight simulators.

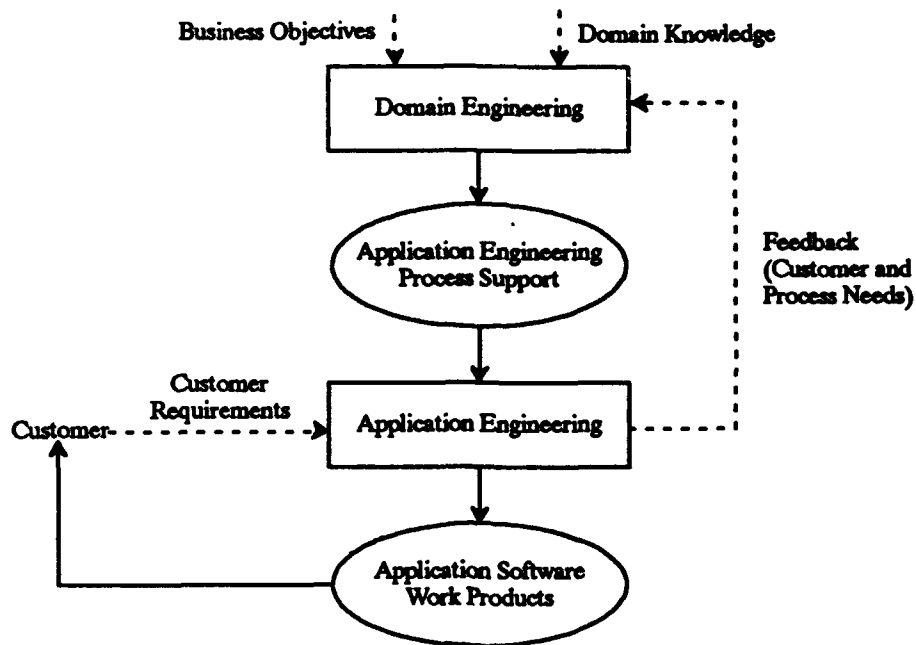


Figure 4-2. Synthesis Process Model

By developing domain process definitions, you can identify the specific process drivers, such as the objectives, alternatives, constraints, and risks, often associated with that particular domain. You can use domain experience and knowledge to determine the candidate solutions available to address those drivers and determine the criteria for tailoring the domain process definition based on predefined process drivers. You can predetermine well-defined variations of activities, methods, practices, tools, and resources to be the optimum strategy for specific process drivers or combination(s) of process drivers.

As a result of working through a set of predetermined questions about process drivers, you can tailor the domain process definition and can appropriately select and allocate methods, practices, tools, and resources to produce an enactable process for a specific application project within the domain. Because application processes will generally vary from the domain process in well-defined ways, it may be possible to automate the activity of tailoring and instantiating the domain process definition to produce a process that is application specific. In addition, no tailoring justification would be necessary unless an application tailored the domain process differently from the predetermined and approved variations.

#### 4.4 SUMMARY

You can adapt the ESP model to represent process engineering from both the organizational and project perspectives. At the organizational level, the process engineer develops reusable assets based on the tactical plan produced by the process improvement spiral that Software Productivity Consortium (1993a) describes. The process engineer proposes and documents asset alternatives and identifies risks to the alternatives. Risk aversion activities justify the asset development that can be achieved during the current cycle, and the information that the aversion activities generates is used to plan and schedule the asset development to be accomplished in the current cycle.

Process assets can take many forms and can be anything that is both useful and reusable in the engineering and enactment of processes on a project. Organizational assets include a baseline of the



existing process capability, standard process architecture, specified process steps, business area analyses, life-cycle model descriptions, and a standard process definition. The process engineer validates and verifies these assets, often through the mechanism of a pilot project. When the appropriate stakeholders complete and approve them, organizational assets come under change control and become part of the process database for use at the project level. The process engineer facilitates asset transfer to the projects, monitors usage, and updates assets based on lessons learned, defects found, and other cost, schedule, quality, and productivity data.

As organizations begin to optimize their organizational process assets, they can begin to develop and automate mechanical tailoring and instantiation guidance based on the specific variations and priorities of process drivers.

This section described the process engineering model, adapted from the conceptual ESP model, by specifically focusing on the model at the organizational level of abstraction. Section 5 continues to elaborate on process engineering with the ESP model by focusing on the project level of abstraction.

## 5. PROJECT PROCESS ENGINEERING

Rather than having a single monolithic process that all projects must use, (organizations) will find that different projects will have differing needs.

Peter H. Feiler and Watts S. Humphrey,  
*Software Process Development and Enactment: Concepts and Definitions*

### 5.1 OVERVIEW

An integral feature of the ESP model is the ability to engineer the best possible process for addressing project needs. When following the ESP model, a project team engineers its process dynamically by continuously documenting and/or tailoring, instantiating, enacting, and improving its project process definition in an evolutionary fashion, as Figure 5-1 shows. Although dynamic, the ESP model specifically provides for orderly and controlled evolution of a baselined project process.

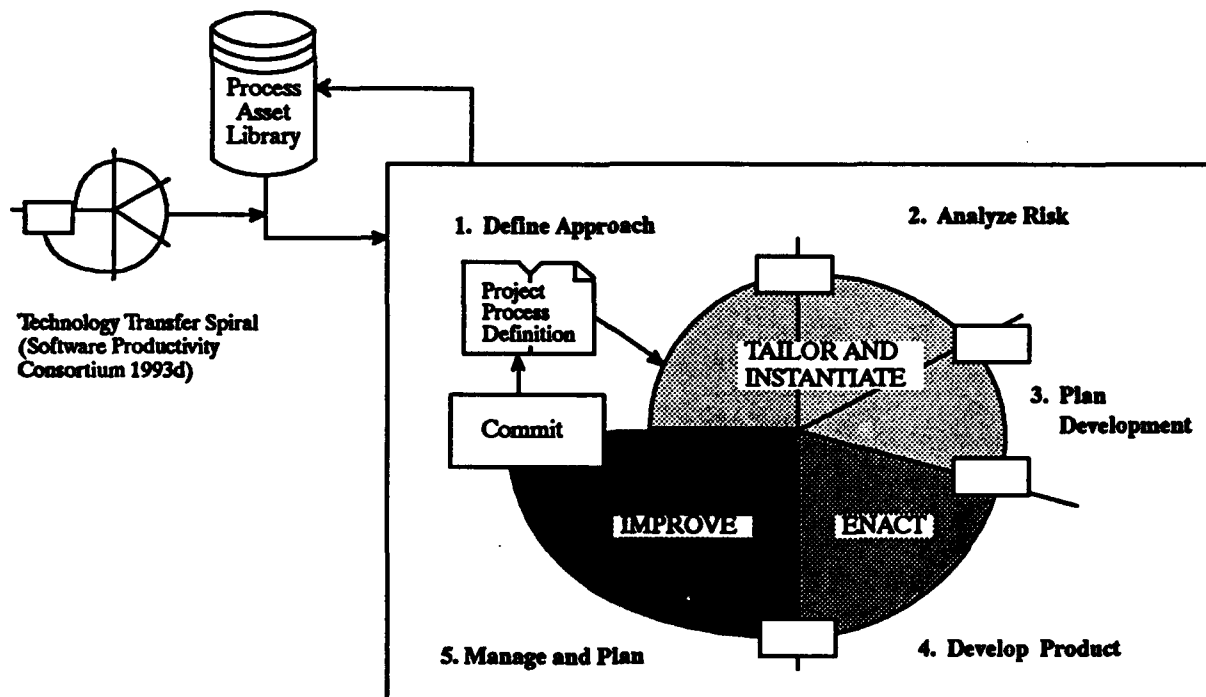


Figure 5-1. Project Process Engineering

Project process engineering emphasizes the identification and continuous evaluation of key project and cycle characteristics with the potential for driving the process in some significant way. In Cycle 1, the process engineer supports the project manager to document a project's process definition either

by defining an overall process from scratch or by tailoring the assets in the organizational PAL to the level of detail appropriate to the current understanding about the project and its unique process drivers. In subsequent cycles, a project tailors and instantiates its project-level process definition by defining it to an enactable level of detail based on cycle process drivers, enacts or performs the cycle process, and evolves the process definition for the remainder of the project based on information gained, lessons learned, progress to date, and early strategies and mitigation decisions.

A project can follow the ESP model to engineer its software development process regardless of the level of organizational process maturity, as the CMM measures it. Although doing so is time and resource intensive, a project can develop its project process definition without having process assets, such as a standard process definition, to use as a foundation. Organizations can abstract and use the project-specific process model, activity specifications, and method descriptions engineered as a result of following the ESP model at an ad hoc or repeatable process maturity level as input to developing a standard process definition.

Experience shows that developing comprehensive standard process definitions can be expensive and time consuming, and it may be more desirable to develop general-purpose process definitions, together with techniques for reusing, tailoring, and enhancing them (Feiler and Humphrey 1992, 4). Projects that are a part of an organization at the defined level of the CMM or above are likely to spend less time and effort engineering their processes because of the availability of a standard process definition or multiple process definitions based on business area. The standard process definition results when the process engineer and the project manager integrate the following process assets into a coherent whole:

- A standard process architecture
- Specified process steps that tools, methods, techniques, and technology transfer mechanisms support
- Specific product life-cycle models

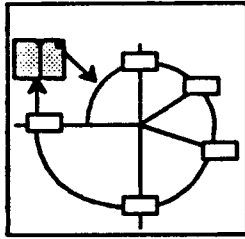
Other process assets that are available to projects that are part of an organization at a higher level of process maturity include a process measures database that is the repository for process and project measures as well as historical technical and management data regarding the use of process assets and an advanced and/or automated organizational software engineering environment. A project at a higher level of process maturity will generally tailor the available standard process definition(s) and other supporting process assets and the historical data into a project process definition that satisfies unique process drivers.

This section focuses on how project process engineering is an integral part of the conceptual ESP model. The ESP model encompasses process engineering actions at the project level by:

- Documenting and baselining a project process definition in the first cycle
- Tailoring and instantiating the process definition for a cycle based on cycle process drivers
- Enacting the instantiated process to produce a product
- Improving a project process definition as a project progresses and the process is enacted

Sections 5.2 through 5.5 discuss each of these topics in more detail.

## 5.2 DOCUMENTING THE PROJECT PROCESS IN THE FIRST CYCLES



The high-level project process definition is one of the spiral planning documents that the first cycles of the project spiral produce.

A project attempts to identify and evaluate those key process drivers that have the potential to affect the entire project, including the product life-cycle perspective that is important to the current project. The project then documents the process at a level of detail that reflects the current understanding about the project and its identified process drivers. A project documents a project process definition as a result of defining the overall process from scratch or of tailoring an existing organizational standard process definition, depending on the process maturity level of the organization.

In the first cycles, a project, which includes appropriate project stakeholders and support from a process engineer, follows the five process steps of the ESP model to:

- Identify and evaluate project-level process drivers.
- Analyze risks and plan for risk aversion strategies.
- Plan the development of the project process definition after averting the identified risks.
- Document the project process by defining it from scratch or by tailoring the standard process definition and other assets in a manner that addresses the project process drivers.
- Continue to plan and manage the project process definition by placing it under configuration control and committing to proceed with the project.

Section 3 briefly describes the five steps of the conceptual ESP model. Sections 5.2.1 through 5.2.5 describe how the first cycles of the project's spiral extend the basic ESP model steps to document the project process definition.

### 5.2.1 IDENTIFY AND EVALUATE PROJECT PROCESS DRIVERS

Process drivers can be the source of process definition or tailoring requirements that a project process must meet. Process drivers tell a project what the key considerations are when attempting to define or tailor the standard process definition to a software development process that will best meet the project's needs. In Step 1 of the first cycles, a project organizes and documents the identified process drivers through the mechanism of the EoS. Specifically, the project documents project-level objectives, success criteria, alternatives, constraints, risks, and other considerations.

One of the most important pieces of information that the EoS documents will be the life-cycle alternative(s) selected for the product that the project is responsible for producing. The project process drivers, which the EoS also documents, can be the source for determining product life-cycle alternatives by helping to identify the key considerations when attempting to define or tailor a life-cycle model that will best meet the project's needs.

For example, if the EoS document shows that the project has specific milestones and firm requirements that the contract imposed and that the contract deliverables follow a linear schedule, then a project may select a waterfall model as a life-cycle alternative. If the contract has a number of loosely defined requirements and expects significant user involvement, then a project may select an evolutionary development life-cycle model may be selected as an alternative. In other environments, the life-cycle alternative(s) might be some combination of life-cycle models.

The waterfall and evolutionary models are examples of perhaps the most well-known software life-cycle models currently used on projects throughout the industry; however, there are several life-cycle models, variants, and combinations that a project can tailor based on its unique needs. For example, the project may only be responsible for the development of a design concept, such as a Conceptual Design Phase contract, and may not need to determine the life cycle for the entire product but only for the part of the life cycle for the current contractual commitment.

The EoS documents candidate life-cycle alternatives, which should describe:

- The key states or milestones associated with each life-cycle alternative
- Potential methods, tools, and/or techniques for reaching each life-cycle state
- High-level resource allocations for each of the life-cycle stages

An organization may have life-cycle descriptions available as assets in the process database. These life-cycle descriptions may be of the more well-known life-cycle models or may be specifically tuned to the key products in a business area product line.

### **5.2.2 ANALYZE PROJECT PROCESS RISKS AND PLAN FOR RISK AVERSION STRATEGIES**

During Step 2 of the first cycles, a project identifies additional project and/or cycle process drivers in the form of risks. Some of these risks will be apparent from the information that the EoS documents; the project can identify other risks as a result of using a risk taxonomy or by analyzing historical data and experience. Risks that have a high overall risk level, based on high probability and cost of occurrence, tend to be the process drivers that place the most significant requirements on the process.

A project can expand the basic Step 2 activities of the conceptual spiral to evaluate risk aversion strategies for all identified process drivers instead of just those that are risks and can commit to a specific process alternative as a result. Strategies for addressing project process drivers may include:

- Defining or tailoring the life-cycle alternative in a way that meets project-specific process drivers
- Defining or tailoring individual process steps by organizing or reorganizing activity sequences and relationships within each step
- Determining high-level resource and/or responsibility allocations
- Identifying potential methods, tools, and/or techniques

A project evaluates how well each strategy addresses the relevant process driver, how the strategy might affect other process drivers, and the costs associated with each strategy.

A project should also carefully consider the technology transfer impact when evaluating risk aversion strategies that involve new technology to the project, such as a new method, tool, or technique. A project should evaluate and commit to the technology transfer the activities and supporting resources needed to introduce the new technology into the project. By not incorporating orderly and defined technology transfer activities as an integral part of the project process definition, a project runs the risk of unsuccessfully addressing the process driver because the project team neither understands nor effectively uses the selected method, tool, or technique alternative.

A project can use *Using New Technologies: A Technology Transfer Guidebook* (Software Productivity Consortium 1993d) when identifying and evaluating strategies that involve new technology. The guidebook provides detailed guidance on identifying and selecting new technologies, as well as defines the process activities necessary to transfer the technology for maximum use and benefit.

### 5.2.3 PLANNING FOR THE DEVELOPMENT OF THE PROJECT PROCESS DEFINITION

In Step 3, a project plans for the development of the project process definition after averting the identified risks according to the risk aversion plan committed to at the end of Step 2. The result of risk aversion activity should justify specific process definition alternatives, such as the product life-cycle alternative.

Note that the project process definition is only one of a series of related spiral planning documents that are planned for development in the first cycles. The other planning documents generally planned for development in the first cycles of the project spiral include:

- **Product Development Plan.** The format of this document may vary depending on the contract requirements and may include the contents of the other planning documents. The contract will determine the amount of detail in this plan. The plan generally includes resource (time, money, personnel, equipment, etc.) estimates, a WBS describing how costs will be collected, and how financial and technical progress will be measured on the project.
- **Other Required Plans.** These plans include other documents that the contract or project requires, including testing plans, configuration management plans, and hardware/software acquisition plans.

Steps 1 and 2 have already developed two other important spiral planning documents, the project EoS and the RMP.

### 5.2.4 DEVELOPING THE PROJECT PROCESS DEFINITION

In Step 4, a project can start to document process at a project level. The resulting project process definition should first identify and define the product life-cycle perspective that is important to the current project and should include estimates of the appropriate life-cycle states that will represent the key development milestones. A project can use a preliminary listing of alternative activities and methods to achieve the estimated life-cycle states. The listing should include any corporate-required activities and methods as well as any project-specific tailoring necessary for these activities and methods.

You develop a project process definition from the ground up or by tailoring an existing organizational standard process definition to meet the requirements that the project-level process drivers and corresponding strategies impose. Project process drivers and resulting strategies drive the documentation

of a project process definition regardless of organizational process maturity level. A project at a low process maturity level organization, which must document its process from scratch, can view process drivers and resulting strategies as process definition requirements. A project at a higher process maturity level organization can view process drivers and strategies as requirements for tailoring a standard process definition to a project-specific one.

Specifically, a project produces the project process definition by:

- Identifying and analyzing product and project process drivers
- Defining product-specific requirements
- Selecting the product life-cycle model appropriate to the project
- Tailoring the product life-cycle model that the project will use to guide the key product's development phases
- Tailoring the appropriate process step specifications from the standard process definition documents to produce the project process definition
- Achieving consensus on the project process definition by all stakeholders
- Baselining the project process definition and including it as part of the spiral planning documents
- Training project staff on the project process definition
- Monitoring and continually improving the project process definition

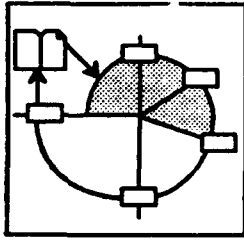
At the end of Step 4, the project process definition should describe the estimated product life cycle, define the supporting process to the level of detail appropriate to the current understanding of the remainder of the spiral, have the commitment of the project stakeholders, and be ready for placement under change control. During subsequent cycles, a project instantiates or creates an enactable process of the process definition for the current cycle based on process drivers you identified for the cycle.

Note that projects that do not have a PAL to use as the foundation for developing the project process definition can use the process product development activity specifications in Appendix B. A project can use the appendix as a source for selecting, tailoring, and integrating process steps to produce the project process definition.

### **5.2.5 BASELINING THE PROJECT PROCESS DEFINITION AND COMMITTING TO PROCEED**

During Step 5, a project places the project process definition under configuration control, along with other planning documents that the first cycles of the project spiral developed. The project updates all planning documents, as necessary, to reflect lessons learned, estimates based on actual data, and updated process drivers. A project should plan the first three steps of the upcoming cycles when updating the planning documents; that is, plan the next cycle planning and risk aversion activities to estimate the situation, analyze and avert risks, and plan development. Brief the appropriate project stakeholders on the results of documenting the project's process definition and commit to pursuing subsequent cycles.

### 5.3 TAILORING AND INSTANTIATING THE PROJECT PROCESS DEFINITION FOR A CYCLE



The first three steps of each cycle tailor and instantiate the project process definition by assigning schedule and resources to produce an enactable process for the project team to follow.

In dynamic process instantiation, a project may interleaf the process development, the instantiation of individual steps of that process, and the enactment of these steps. The dynamic case permits a project to start enactment of processes before their definition is complete (Feiler and Humphrey 1992). The ESP model supports dynamic instantiation by tailoring and instantiating the project process definition one cycle at a time. The project identifies and tailors the development activities for producing the appropriate product or part of the product in the current cycle and then instantiates those activities to an enactable level of detail. In the context of the ESP model, a project refers to the artifact that results from tailoring and instantiating the project process definition in each cycle as the cycle plan rather than the process plan.

Specifically, the project tailors and instantiates the project-level process definition, produced during the first three steps of the current cycle by:

- Determining and evaluating cycle-level process drivers that will place requirements on process alternatives for the cycle
- Analyzing process risks and planning for risk aversion strategies
- Documenting the current cycle process by tailoring it based on the selected alternative and then instantiating it to an enactable level of detail

Sections 5.3.1 through 5.3.3 further describe these process tailoring and instantiation activities, which map to the conceptual ESP model activities.

#### 5.3.1 DETERMINE CYCLE PROCESS DRIVERS AND ALTERNATIVES

Cycle process drivers can be the source of tailoring and instantiation requirements that project process must meet. Current cycle process drivers are usually a decomposition of the project process drivers that specifically affect the current cycle or come from the previous cycles. During Step 1 of the current cycle, a project uses the plans, metrics, and status reports that the previous cycles produced or updated to identify cycle process drivers. As a result, a project updates the EoS that the first cycle produced with the current cycle objectives, success criteria, alternatives, constraints, risks, and other considerations.

One of the main sources of cycle drivers is the project process definition that the first cycles of the spiral developed. Remember that the project process definition defines the product life-cycle perspective that is important to the project and includes estimates of the appropriate life-cycle states that will represent key development milestones. The project process definition also includes a listing of alternative activities and methods that a project may use to achieve the estimated life-cycle states.



During Step 1 of the current cycle, a project should identify what the key development objectives will be for the cycle, that is, the product life-cycle state that the project wishes to reach as a result of the current cycle. A project should also reevaluate the listing of alternative process steps and supporting activity specifications, methods, and tools to use to achieve the desired life-cycle state, such as a specific design method or the decision to prototype the user interface before producing a detailed design document.

A project should identify and further define appropriate cycle process alternatives in terms of:

- The set of appropriate development, planning, and support process steps to follow in that cycle
- Any tailoring to do to those steps
- How to perform each step, including the use of methods, practices, and tools
- What each step will accomplish or produce
- Who will perform the step
- How much time and effort the project thinks the step will take

A project tends to have the most flexibility for addressing key process drivers in the first cycles. The decisions and commitments a project makes and the actions it takes as the project progresses create a legacy that may place restrictions on process alternatives in later cycles. The greater the legacy, the less process flexibility a project may have. In some cases, however, the best strategy may be to wait until later in the process before evaluating and committing to a specific process alternative. For example, a project may identify one or more potential design methods but may not want to commit to a specific method until the impact of requirements is more clearly understood.

### **5.3.2 ANALYZE CYCLE PROCESS RISKS AND PLAN FOR RISK AVERSION STRATEGIES**

During Step 2, a project follows the standard ESP activities to:

- Identify risks to the process alternatives.
- Analyze strategies for averting those risks, such as tailoring the process steps in a specific way.
- Commit to a specific risk aversion plan.

The results of executing the risk aversion plan should justify a specific process alternative for the cycle. The project team and other stakeholders should agree that the selected process alternative represents the best solution for meeting cycle objectives and success criteria, averting cycle risks, and addressing other cycle process drivers.

### **5.3.3 DOCUMENT THE CYCLE PROCESS**

During the Plan and Schedule activity in Step 3, a project documents the cycle process to an enactable level of detail by producing a cycle plan. The cycle plan should describe the process alternatives selected to address the cycle process drivers and should be consistent with the framework that the

project-level process definition provided. If a project cannot produce a cycle plan that is generally consistent with the project-level process definition, then the project should consider the inconsistency as a risk, evaluate any potential impact on subsequent cycles, and commit to evolving the project-level process definition as appropriate.

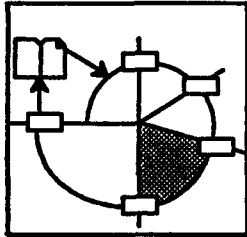
A project tailors and instantiates the process for the current cycle when:

- The project team and other stakeholders identify and approve cycle process drivers, including cycle objectives and success criteria.
- A project addresses the cycle process drivers by:
  - Selecting process alternatives from the listing documented at a high level in the project process definition
  - Defining and/or tailor previous alternatives as appropriate to mitigate any identified risks
  - Gaining commitment from all stakeholders
- A project documents the cycle plan which details the:
  - Size, cost, schedule, and errors estimated for the product or part of the product to be developed during the cycle
  - Size, cost, schedule, and error data collection mechanisms
  - Risk monitoring mechanisms
  - Selected methods, practices, tools, and other process alternatives for supporting the cycle
  - Cycle activities required to meet cycle process drivers and alternatives and how the activities relate to each other
  - Specified process steps that describe how to perform the cycle activities and that you define or tailor as required to meet cycle objectives and success criteria, accommodate cycle alternatives, satisfy cycle constraints, and avert cycle risks
  - Allocation of resources and responsibilities to support all cycle activities
  - Work packages for the key cycle activities
- The project stakeholders have approved the cycle plan and work assignments.
- A project places the cycle plan under change control and distributes it to all stakeholders.
- The project manager opens cycle work packages and authorizes charges to them.

A simple tool that a project can use to represent an instantiated process graphically is a PERT chart, which can represent the process in terms of the set of cycle activities and supporting activity sequence,

estimated durations, start and end dates, and resources. Some PERT chart tools capture information, such as fields to capture the WBS code and cost data; free text fields for brief descriptions of activities, artifacts, and/or supporting methods; and historical data and metrics.

#### 5.4 ENACTING THE CYCLE PROCESS



Step 4 of the cycle enacts the process definition to produce the product necessary to meet cycle objectives.

A project enacts the cycle process by following the Step 4 activities to:

- Perform to the cycle plan to develop and verify the product or part of the product.
- Monitor and review the cycle activities as a project performs them.
- Analyze the results for technical merit.
- Verify results against cycle objectives and success criteria.

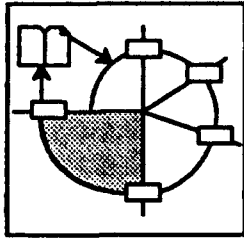
In Cycle 1, the product that enacting the process produces is generally the set of project planning documents. In subsequent cycles, the enacted process should produce the product or part of the product necessary to meet the cycle objectives and success criteria.

At a minimum, a project should collect the following basic project data as an integral part of enacting the cycle process:

- Actual total labor effort for each planned and scheduled cycle activity
- Actual number and type of product components (for example, modules, documentation) produced
- Actual size of each product component
- Actual number, severity, and source of errors found during verification activities
- Actual risk increase or reduction of each high-priority risk and of the overall project risk
- Product change data
- Lessons learned

A project can use the enacted cycle process together with lessons learned and the data collected by monitoring and reviewing the cycle activities are used to review and subsequently evolve the process for the remainder of the project. A project can use the *Software Measurement Guidebook* (Software Productivity Consortium 1992b) for specific guidance on defining, estimating, and collecting size, cost, schedule, and error data.

## 5.5 IMPROVING THE PROJECT PROCESS



Step 5 of the cycle includes analyzing actual data and lessons learned to correct and evolve the project process definition.

Feiler and Humphrey (1992, 5) state that, as projects evolve and members of project teams gain experience with the process, there will be many ideas for improvement; therefore, it is important to establish mechanisms to obtain continual user feedback to guide process repair and evolution. The ESP model provides for these mechanisms in Step 5 by specifically including process activities intended to help a project review the process to date and to plan and manage process evolution. It is important for a project to control process review and evolution carefully because the results of these activities often place instantiation requirements on subsequent cycles or may even require rework of previous activities.

A process can perform process review and subsequent evolution within the current cycle if evolving the process affects the current cycle only. As a project enacts the current cycle process in Step 4, it may discover the need to reallocate resources, replan schedules, or reassess risks to meet cycle objectives and success criteria. If the impact of addressing the new needs affects only the current cycle activities, then a project can quickly review the process drivers, evolve the cycle process accordingly, and baseline the evolved cycle process.

After meeting cycle objectives and success criteria and baselining the product or part of the product produced during the current cycle, the project reviews progress to date, including:

- The basic project data collected when enacting the cycle process
- How well the enacted process accomplished cycle objectives and success criteria
- How closely the project followed the process as instantiated for the cycle
- Identified process strengths, weaknesses, and defects
- Newly discovered or better understood objectives, alternatives, constraints, risks, or other process drivers

As a result of the review, a project can evolve its project process definition to correct defects or weaknesses or to meet new process drivers by:

- Updating the EoS to document new or better understood process drivers
- Updating engineering and project procedures to reflect lessons learned, updated process drivers, or changes in development alternatives
- Updating the project process model and schedule, staff, and budget estimates as necessary to reflect changes based on actual cycle data

- Getting approval of the evolved version of the project process definition from stakeholders
- Baselining the evolved version of the project process definition

When enacted, a review of the process to date may place significant evolution requirements on the project's process for the remainder of the project. For example, a project may:

- Uncover or clarify process drivers that affect subsequent cycles, such as actual labor hours remaining to support the project.
- Uncover a process driver that requires rework of already enacted activities, such as a failed quality metric, a misunderstood requirement, or a fatal design flaw.
- Evolve the project process definition in a way that affects the remainder of the project, such as updating a policy or procedure.

After a project has successfully met all the project-level objectives and success criteria, it can take advantage of the Step 5 process activities to review the final project process definition and enactment data as a whole to identify trends, statistics, and improvements to use as input to the next project or to evolve organizational process assets.

### 5.6 SUMMARY

Using the ESP model results in evolutionary process engineering at the project level by:

- ***Defining a Project's Process in the First Cycle.*** A project identifies and analyzes its project-level process drivers to define the project-level process definition or to tailor the organizational standard process definition as required to address the drivers. A project physically documents the project-level process definition in Cycle 1 through a subset of the spiral plan documents. A project defines the project-level process definition to the level of detail appropriate to the current understanding of the known process drivers and the remainder of the spiral activities. After the stakeholders approve it, it comes under change control.
- ***Tailoring and Instantiating the Project Process for a Cycle.*** A project tailors and instantiates the process definition for the current cycle as required to address cycle process drivers. Cycle process drivers include cycle objectives, success criteria, constraints, process step alternatives, and risks. Cycle process drivers also incorporate the legacy from previous cycles. A cycle plan documents all the information required to enact the cycle process. If the cycle plan is not generally consistent with the project-level process definition, then the project should consider the inconsistency as a risk, evaluate any potential impact on subsequent cycles, and commit to evolving the project-level process definition as appropriate. When the stakeholders define and approve the cycle process, it comes under change control.
- ***Enacting the Cycle Process.*** A project enacts the cycle process in Step 5 of the current cycle by performing to the cycle plan, monitoring and reviewing the cycle activities as they occur, analyzing the results for technical merit, and verifying results against cycle objectives and success criteria.
- ***Evolving the Project Process.*** A project reviews previous cycle data and evolves the process definition for the remainder of the spiral based on information gained, lessons learned,

progress to date, and early strategies and mitigation decisions. Process review and evolution often place instantiation requirements on subsequent cycles. In some cases, a project can quickly evolve and instantiate the process definition for the current cycle if the impact of the evolution only affects that cycle. In all cases, the stakeholders approve the evolved cycle or project process definition and place it under change control.

After a project has successfully met all of the project-level objectives and success criteria, it can take advantage of the Step 5 process activities to review the final project process definition and enactment data as a whole to identify trends, statistics, and improvements to use as input to the next project or to evolve organizational process assets.

*This page intentionally left blank.*

# APPENDIX A. EVOLUTIONARY SPIRAL PROCESS ACTIVITY SPECIFICATIONS

## A.1 OVERVIEW

This section defines the key activities essential to generating a process using the ESP model. Table A-1 gives a complete list of the ESP activities specified in this section.

Table A-1. Evolutionary Spiral Process Activities

Step	Activity Name
1. Understand Context	Define Approach Develop/Update Estimate of the Situation Review Context
2. Analyze Risks	Perform Risk Analysis Review Risk Analysis Plan Risk Aversion Commit to Aversion Strategy
3. Plan Development	Execute Risk Aversion Review Alternative Plan and Schedule Commit to Plan
4. Develop Product	Develop and Verify Product Monitor and Review Review Technical Product
5. Manage and Plan	Product Change Control Review Progress Update Spiral Planning Documents Commit to Proceed

The format for the activity definitions uses an enhanced set of entry-task-validation-eXit (ETVX) attributes. Figure A-1 depicts an activity based on the ETVX paradigm (Radice and Phillips 1988). The entry and exit criteria restrict products from entering or leaving the activity unless they are at a prescribed state of completeness. The task(s) performs the real work of the activity. Validation evaluates the work product to determine whether it satisfies customer needs.



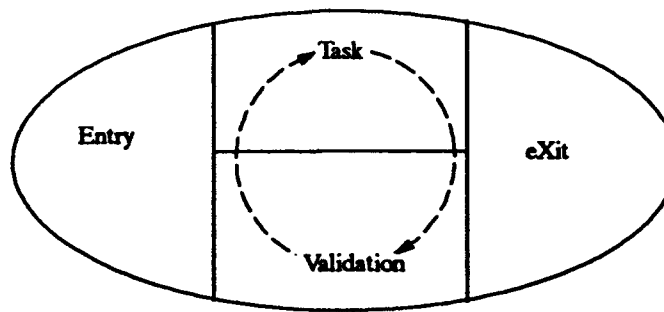
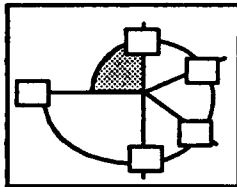


Figure A-1. Entry-Task-Validation-eXit Notation

An activity can be performed as soon as all its entrance criteria are met. These entrance criteria should include a check of the state of its inputs. The typical operation of an activity defined using this notation is to transform input products into the next level of completeness by performing a task. A task is repeated as necessary until it is satisfactorily completed and the exit criteria are met.

The following format is used for the activity definitions in this appendix.



This section maps the activity to the ESP model by identifying the step where the activity begins.

#### **Overview**

This section gives a brief description of the activity and describes the tasks to be performed.

#### **Performers**

This section identifies who will execute the activity. Performers include project stakeholders, technical leads, project managers, risk analysts, contract managers, engineers, quality managers, users, customers, configuration managers, and corporate managers.

#### **Inputs**

This section identifies what artifacts will be used during the performance of the activity. It is not necessary for all the inputs to exist to begin the activity. It is possible for only a subset of inputs to exist during the first iteration of the activity; as more inputs become available, subsequent passes are made through the activity.

Supporting documents and spiral planning documents are inputs to many of the activities described in this appendix. These inputs do not refer to a single artifact, but rather to a set of documents. Specifically, supporting documents include the following:

- Contract
- Statement of work (SOW)
- Documents describing the customer:

- Target environment
- Policies, procedures, standards, and regulations
- Requirements
- Internal infrastructure data, including:
  - Internal culture and investments
  - Organizational process definition, if available
  - Historical engineering data, if available
  - Current software engineering environment and support

The spiral planning documents are a set of related project planning documents that include the following:

- Project EoS
- RMP
- Cycle plan
- Product development plan
- Project process definition
- Other required plans, such as a configuration management plan, testing plan, or acquisition plan

***Outputs*** This section identifies what artifacts will be generated as a direct result of performing the activity.

***Entrance Criteria*** This section identifies prerequisites that the input artifacts must satisfy to begin the activity.

***Exit Criteria*** This section states what conditions the output artifacts must meet before the activity is considered complete. It is assumed that each activity satisfies the following exit criteria:

- Each artifact generated by the activity has been baselined, verified, and approved.
- Each artifact generated by the activity conforms to the organizational policies, standards, and procedures.
- Each artifact generated by the activity has collected process, risk, and quality assurance information.

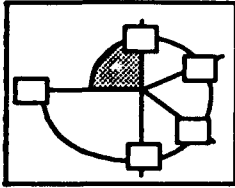
***Measurables***

This section describes the measurables to be collected during the execution of the activity. The measurables are used to monitor activity progress, to aid in estimating the effort required to complete the activity, and to improve the process.

Note that every activity measures total activity labor effort:

- Actual calendar start and end dates
- Team members who performed the activity
- Total labor hours expended on the activity

## A.2 DEFINE APPROACH



This activity begins in Step 1, Understand Context.

### **Overview**

The approach definition establishes the ground rules for measuring progress at the spiral and cycle levels. Identify, collect, and analyze input artifacts to define process drivers. If desirable, interview the stakeholders for additional insights and information. Define the following process drivers with enough precision to establish objective measures for determining whether the spiral or cycle is meeting expectations or whether the expectations are infeasible:

- **Stakeholders.** Who has a vested interest in the success of the spiral or cycle.
- **Objectives.** What is to be accomplished during the spiral or cycle.
- **Alternatives.** Different ways to meet the objectives.
- **Constraints.** Limitations on alternatives.

### **Performers**

Technical lead, project manager, risk analyst, and contract manager

### **Inputs**

- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### **Outputs**

The output of this activity is the documented approach definition.

### **Entrance Criteria**

- A signed contract or authorization to proceed exists.
- Resources are allocated to enact the first cycle activities.
- All stakeholders have approved updated spiral planning documents.
- Management gives authorization to proceed in current cycle.

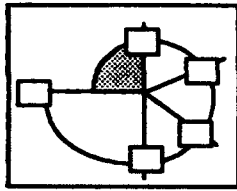
### **Exit Criteria**

- Spiral-level and cycle-level process drivers are defined in terms of objectives, alternatives, constraints, risks, and other considerations to the level of detail appropriate to the current level of understanding.
- The legacy inherited from previous cycles has been analyzed, and appropriate changes have been made to the approach definition.

### **Measurables**

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

### A.3 DEVELOP/UPDATE ESTIMATE OF THE SITUATION



This activity begins in Step 1, Understand Context.

#### **Overview**

An EoS is performed during the first cycles of a spiral and updated during subsequent cycles. At a minimum, the EoS should identify, analyze, and document the scope of the spiral, the spiral's process drivers in the form of objectives, alternatives, product and process assumptions, and the assets available for meeting the spiral objectives (Charette 1989).

The following process drivers should be specifically identified, analyzed, and documented in the EoS:

- Mission and history
- All identified stakeholders and stakeholder expectations
- Spiral objectives
- Current cycle objectives
- Inherited decisions and assumptions
- Characteristics of known requirements and operations
- Known constraints
- Factors that may affect successful completion of the spiral and current cycle
- Comparison of possible courses of action

Process drivers can be identified from several sources, such as the contract and SOW; insights about and from the client; historical project plans and budgets; client and business area policies, procedures, and standards; and interviews with key stakeholders at all levels.

#### **Performers**

Technical lead, project manager, and risk analyst

#### **Inputs**

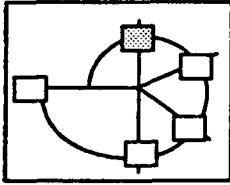
- Approach definition
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

#### **Outputs**

The output of this activity is the draft EoS.

<b><i>Entrance Criteria</i></b>	The entrance criterion for this activity is that the spiral or cycle process drivers have been identified and analyzed.
<b><i>Exit Criteria</i></b>	The exit criterion for this activity is that an EoS has been updated to the level of detail appropriate to the current level of understanding about the spiral or cycle process drivers.
<b><i>Measurables</i></b>	The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.4 REVIEW CONTEXT



This activity begins in Step 1, Understand Context.

### **Overview**

To proceed to Step 2, first obtain firm agreement on what to pursue for this stage of development. Consensus should be reached on the objectives, constraints, and alternatives documented in the EoS for the spiral or current cycle. Specifically, this activity should ensure that:

- All stakeholders are identified and their expectations clearly documented.
- Objectives are clearly documented and represent reasonable winning conditions for all stakeholders.
- Current cycle-level objectives are derived from or are refinements of the spiral-level objectives.
- Proposed alternatives adequately address objectives and constraints.
- All the stakeholders agree to pursue the spiral and current cycle.

Make sure that all key stakeholders review and approve the EoS. Although the EoS does not need to circulate widely outside of the immediate team, make sure that the senior manager reviews, corrects, and signs off on the EoS to ensure that the documented objectives are clear and correct (Charette 1989).

The EoS may be modified as a result of reaching agreement. Document all changes with the change rationale in the meeting minutes and distribute the minutes to all attendees.

### **Performers**

All project stakeholders, such as technical lead, engineer, contract manager, quality manager, user, project manager, risk analyst, and customer

### **Inputs**

- A draft EoS
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### **Outputs**

- Meeting minutes
- The approved EoS

### **Entrance Criteria**

- The spiral EoS document is drafted or updated to address the current cycle.

- All individuals with an interest in the success of the spiral activities or their representatives are participating.

***Exit Criteria***

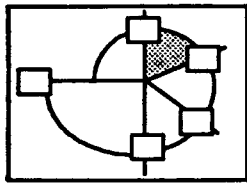
The exit criterion for this activity is consensus to proceed.

***Measurables***

- The activity labor effort (see Section A.1)
- The number and type of changes to the EoS



## A.5 PERFORM RISK ANALYSIS



This activity begins in Step 2, Analyze Risks.

### *Overview*

The Perform Risk Analysis activity is initiated by comprehensively identifying potential spiral or current cycle risk items. Examine the objectives with respect to alternatives, constraints, organizational and project assets, and other process drivers, and identify what can go wrong. Examine these unsatisfactory outcomes and the effect on both the current cycle and spiral success criteria; that is, if a significant risk is identified for the current cycle, trace it back to the spiral-level objectives and success criteria to determine whether the risk may affect the spiral as well as the cycle. To help identify typical spiral risks, use risk taxonomies such as in Boehm and Ross (1989, 117), U.S. Air Force (1988), and Charette (1990, 216–59).

You analyze risks once they are identified by categorizing them and determining risk likelihood and consequence. Estimate the chance of potential loss (or gain) and the consequence (or benefit) of the risk situations previously identified. Analyze the risk situations independently of one another. Show any uncertainties in the estimates.

Charette (1989, 120–24) and the U.S. Air Force (1988) describe several approaches for developing measurement scales. When risks are quantified, compare, rank, and communicate them to appropriate stakeholders. Determine the high-priority risks by identifying those with the highest calculated overall risk factor degrees based on the probability and cost of occurrence. It may be helpful to use a visual representation of risks as a communication and documentation tool. One risk visualization method, the iso-risk contour, is explained in Charette (1989).

After completion of the risk analysis, a risk evaluation is performed to identify risk aversion strategies and examine the impact of the risk aversion strategy on each high-priority risk item. Any risk aversion strategy identified should reduce the cost or probability of risk occurrence to an acceptable level. An option is to establish a risk referent, a measure against which to determine the amount of risk acceptable for each individual and overall spiral risk. Try to define a strategy that can reduce the risk to an acceptable level. In one case, this may require a very active approach; in another, it may mean accepting the risk. Define the strategy activities to reduce the cost of occurrence of a risk and the probability of occurrence of a risk.

It is possible for risk aversion strategies to introduce new risks that will detract from the anticipated benefits or negatively affect other risks; therefore, address the impact of a risk aversion strategy on other risks and process drivers.

In general, consider the following when defining risk aversion strategies:

- Is the strategy feasible?
- Does the strategy reduce risk to an acceptable level?
- Will the strategy negatively affect another risk?
- What is the potential impact of new risks, if any, introduced by the strategy?
- Does the strategy support cycle or spiral objectives and success criteria?
- Are the tactics and means for implementing the strategy consistent with cycle or spiral constraints?
- Is the strategy cost-effective?

Use *Using New Technologies: A Technology Transfer Guidebook* (Software Productivity Consortium 1993d) when identifying and evaluating strategies that involve new technology. This guidebook provides detailed guidance on identifying and selecting new technologies and defines the process activities necessary to insert the technology for maximum use and benefit.

Document the results of this activity in the draft RMP. The plan is enhanced and matured as you conduct the remainder of the Step 2 activities.

**Performers**

Risk analyst, technical lead, and project manager

**Inputs**

- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

**Outputs**

The output of this activity is a draft RMP that documents:

- A list of identified spiral and current cycle risk items
- An estimate of the probability and cost of occurrence for each of the risk items
- A ranking of risk items
- Risk aversion strategies for each risk item ranked as high priority

**Entrance Criteria**

- Stakeholders have reviewed and approved the EoS document.
- Stakeholders have reached consensus to proceed with the current cycle.

**Exit Criteria**

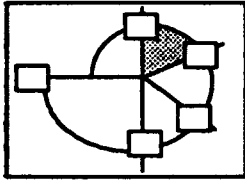
- The project has analyzed the project and current cycle objectives with respect to the alternatives and constraints and has identified the risk items (what can go wrong).

- The project has considered the impact of the identified risk items on the project and current cycle success criteria.
- The project has estimated the likelihood of occurrence (probability of successful or unsuccessful outcome) for each risk item.
- The project has estimated the potential damage for each risk item (gain or cost associated with the outcome).
- The project has calculated an overall degree of risk.
- The project has stated explicitly any uncertainty in the estimates.
- Alternative strategies for averting the high-priority risks have been elaborated, ranked, and analyzed.
- The interaction among the individual risks has been evaluated, and additional risks or risk aggravation has been introduced as a result of each strategy being identified.
- The risk aversion strategies for each high-priority risk have been documented in the draft RMP, including any backup data or analysis necessary to support the findings.
- The amount of risk that is acceptable for each individual and overall spiral risk has been identified.

***Measurables***

- The total activity labor effort (see Section A.1)
- The total number of spiral-level risk items
- The total number of current cycle risk items
- The total number of high-priority risk items

## A.6 REVIEW RISK ANALYSIS



This activity begins in Step 2, Analyze Risks.

### **Overview**

The Review Risk Analysis activity is an opportunity for the team to review and comment on the results of the risk identification, analysis, and evaluation activities. Submit the draft risk management plan to the rest of the team for review. A team meeting is a useful mechanism for discussing team comments and changes. The team should reach consensus on the identified risks, corresponding probability and cost of occurrence, high-priority risk items, possible aversion strategies, and potential impact of aversion strategies.

If the draft RMP is modified, document all changes with the change rationale in the meeting minutes and distribute the minutes to all attendees.

### **Performers**

Project manager, risk analyst, technical lead, and engineer

### **Inputs**

The input to this activity is the draft RMP.

### **Outputs**

- Meeting minutes
- Approval of the draft RMP

### **Entrance Criteria**

- A list of all cycle and spiral risk items that may be identified has been produced, given the current level of spiral and cycle understanding.
- The probability and cost of occurrence for each risk item have been estimated.
- The highest priority risks, based on overall risk factors, have been determined.
- Risk aversion strategies for each high-priority risk item have been identified, and the impact of the strategy on other risks and process drivers has been analyzed.
- The team members have received the draft RMP for review.

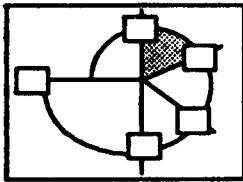
### **Exit Criteria**

- The team members reached consensus regarding any changes to the draft RMP.
- All agreements or changes to the draft RMP have been documented in the meeting minutes, and the minutes have been distributed to all attendees.

### **Measurables**

- The total activity labor effort (see Section A.1)
- The number of changes to the RMP

## A.7 PLAN RISK AVERSION



This activity begins in Step 2, Analyze Risks.

### **Overview**

Estimate and evaluate the high-level cost and schedule for each risk item and determine the best strategy for each. Try to plan risk aversion strategies so that their results are available in time to plan for the development activities that depend on them. Ideally, the team is involved in this activity to help evaluate and recommend the best risk aversion strategies.

It is possible to manage and plan the risk reduction strategy through a new risk reduction cycle or subspiral. Make certain that the cycle or subspiral is planned so that any development that depends on the results is not started until after integration is complete.

Document the results of this activity in the draft RMP. The plan is enhanced and matured as you conduct the remainder of the Step 2 activities.

### **Performers**

Project manager, risk analyst, technical lead, and engineer

### **Inputs**

- The approved draft RMP, including:
  - A list of identified spiral and current cycle risks
  - The probability of occurrence for each risk item
  - The cost of occurrence for each risk item
  - A ranking of risk items
  - The risk aversion strategies for each high-priority risk item
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### **Outputs**

The output of this activity is the draft RMP updated with the plans for the risk aversion strategies.

### **Entrance Criteria**

The entrance criteria for this activity are team review and consensus on the draft RMP to date.

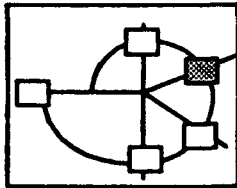
### **Exit Criteria**

- The cost and schedule associated with each risk aversion strategy have been estimated.
- A recommended strategy for each risk item has been selected.
- Recommendations to change the EoS by elaborating the alternatives, suggesting changes to the constraints, or defining new objectives have been made.

### **Measurables**

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.8 COMMIT TO AVERSION STRATEGY



This activity begins in Step 2, Analyze Risks.

### Overview

The Commit to Aversion Strategy activity is a mechanism for formally briefing all stakeholders on the contents of the RMP. As a result of the briefing, the consensus and commitment to the risk aversion strategies recommended for each high-priority risk item should be reached. If consensus is not reached and commitment not secured or if the risk aversion strategy committed to is not the one recommended, then the risk activities may need to be repeated, as appropriate.

Consensus on any updates to the objectives, success criteria, constraints, and alternatives documented for the spiral or current cycle in the EoS must also be reached.

If modifications are made to the draft RMP or the EoS, document all changes with the change rationale in the meeting minutes and distribute the minutes to all attendees.

### Performers

All project stakeholders, such as technical lead, engineer, contract manager, quality manager, user, project manager, risk analyst, and customer

### Inputs

- An updated draft RMP, including:
  - A list of identified spiral and current cycle risks
  - The probability of occurrence for each risk item
  - The cost of occurrence for each risk item
  - A ranking of each risk item
  - The risk aversion strategies for each high-priority risk item
  - The cost and schedule for each risk aversion strategy
  - The recommended strategies
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### Outputs

- The approved RMP
- Meeting minutes

***Entrance Criteria***

- The best candidate aversion strategy for each identified risk has been recommended.
- Recommendations to change the EoS by elaborating the alternatives, suggesting changes to the constraints, or defining new objectives have been made.
- All individuals with an interest in the success of the spiral or their representatives have received the draft RMP for review.

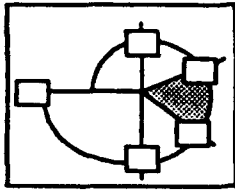
***Exit Criteria***

- Stakeholders have reached consensus on a risk aversion strategy for each high-priority risk item.
- Stakeholders have approved the RMP.
- Stakeholders have approved any changes to the EoS.

***Measurables***

- The total activity labor effort (see Section A.1)
- The number of changes to the RMP
- The number of changes to the EoS

## A.9 EXECUTE RISK AVERSION



This activity begins in Step 3, Plan Development.

### *Overview*

The Execute Risk Aversion activity performs tasks supporting the risk aversion strategy. These may include prototyping, simulation, surveys, comparative evaluations, evolutionary development, and other appropriate techniques. Complicated or long-term risk aversion activities may be spun off into their own spirals to be managed independently from mainline development. The results of the risk aversion activities should justify a particular development alternative, such as a specific method, strategy, or approach. Assess how this development alternative affects the process for the cycle and for the remainder of the spiral. After a development alternative is selected, you should plan and schedule technical activities in detail. Document all relevant information by updating the RMP.

### *Performers*

Project manager, risk analyst, technical lead, and engineer

### *Inputs*

The input to this activity is the approved RMP.

### *Outputs*

The output of this activity is the development alternative for averting each high-priority risk item documented in an updated version of the RMP.

### *Entrance Criteria*

The entrance criteria for this activity is that consensus on a risk aversion strategy for each high-priority risk item has been reached.

### *Exit Criteria*

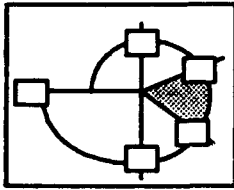
- The tasks supporting the risk aversion strategy have been performed.
- A development alternative has been determined and justified as a result of performing the risk aversion strategy.
- The effect of the development strategy on the cycle and spiral process has been assessed.
- The RMP has been updated, as appropriate.

### *Measurables*

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).



## A.10 REVIEW ALTERNATIVE



This activity begins in Step 3, Plan Development.

### ***Overview***

In the Review Alternative activity, the results of the Execute Risk Aversion activity are presented to appropriate stakeholders and their commitment is solicited to the development alternative or approach selected as a result of performing the risk aversion strategy. If the development alternative selected as a result of the risk aversion activities is not committed to by senior management, then the risk activities may need to be repeated, as appropriate.

### ***Performers***

All project stakeholders, such as technical lead, engineer, contract manager, quality manager, user, project manager, risk analyst, and customer

### ***Inputs***

- An updated RMP
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### ***Outputs***

- Meeting minutes
- The approved RMP

### ***Entrance Criteria***

- A development alternative has been determined and justified as a result of performing each risk aversion strategy.
- All individuals with an interest in the success of the spiral or their representatives are participating.

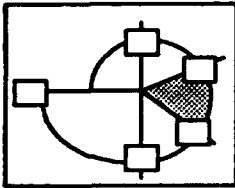
### ***Exit Criteria***

- Consensus has been reached on the development alternative or approach selected as a result of performing the risk aversion strategy.
- The updates to the RMP have been approved.

### ***Measurables***

- The total activity labor effort (see Section A.1)
- The number of changes to the RMP

## A.11 PLAN AND SCHEDULE



### Overview

This activity begins in Step 3, Plan Development.

The main output of the Plan and Schedule activity is a detailed and enactable draft cycle plan. The project's process definition document of the spiral planning documents is instantiated to address cycle objectives, success criteria, constraints, development alternatives, and risks and to incorporate the legacy inherited from previous cycles.

In the Plan and Schedule activity, technical activities are identified and defined, and methods, practices, or tools to be used to complete each task are specified. Include in the activities sufficient interfacing events (such as interface control working groups, management reviews, in-process reviews) to coordinate this cycle's activities with other cycles as well as to satisfy customer control and monitoring needs. Also include ongoing support activities such as configuration control, quality assurance, and documentation.

Although basic management functions of planning, monitoring, and controlling are performed continuously, you should identify, organize, and allocate resources for specific cycle development and development support activities after the cycle risks have been averted. The main output of the Plan and Schedule activity is a cycle plan that plans the development of the product or part of the product that will be produced in the current cycle as well as the development process that will be used.

Your cycle plan can be documented as part of your spiral planning documents, or it can be a standalone document. The plan should:

- Establish development goals and associated development success criteria that support the current cycle objectives
- Estimate size and scope for the development to be accomplished in the current cycle, including:
  - The number and type of product components (e.g., modules, documentation) to be produced as a result of enacting the cycle plan
  - The size of each product component
  - The number, severity, and source of errors likely to be found during verification activities
- Identify the activities or methods to be performed in the current cycle
- Define/redefine activities, methods, and supporting artifacts, if necessary

- Sequence the activities if not sequenced by the selected method
- Estimate development cost and schedule for each cycle activity and allocate resources
- Select and allocate supporting tools
- Define work packages, or the lowest WBS level, for the key activities defined for the current cycle

In addition, remember to:

- Include ongoing support activities in the cycle plan, such as configuration management, quality assurance, and documentation.
- Address any customer requirements.
- Comply with customer policies, procedures, standards, and regulations if necessary.
- Comply with the organizational process definition, if available and appropriate.
- Use any available historical planning and engineering data for estimating purposes.
- Take advantage of in-house methods, tools, training, and support.

***Performers***

Project manager, technical lead, risk analyst, and engineer

***Inputs***

- The approved RMP
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

***Outputs***

The output of this activity is the draft cycle plan instantiated to an enactable level of detail.

***Entrance Criteria***

- Commitment to the development alternative has been secured.
- Spiral status and engineering data from previous cycles has been collected and analyzed, and the spiral planning documents have been updated accordingly.

***Exit Criteria***

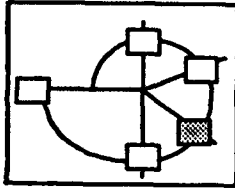
- Cycle activities have been identified and documented in the cycle plan.
  - The cycle activities are consistent with the objectives of the cycle and satisfy the cycle success criteria as defined in the current version of the EoS.

- The cycle activities are consistent with the development alternative as documented in the current version of the RMP.
- The cycle activities are consistent with the organizational process definition, if any or if appropriate.
- Cycle activities have been defined or tailored from the organizational process definition, if any, as required to meet the cycle objectives and success criteria, accommodate cycle alternatives, satisfy cycle constraints, and avert cycle risks.
- The dependencies between activities and the dependencies of the tasks within each activity have been documented.
- Costs have been estimated, and work packages or the lowest WBS level has been opened for the key cycle activities.
- Durations have been estimated for each activity.
- Resources have been allocated to each activity.
- Intermediate milestones and performance measurements have been defined for each cycle activity.
- The monitoring and review activities have been scheduled to accumulate data for and analyze the status of cycle activities.
- Sufficient interfacing events have been scheduled, such as interface control working groups, management reviews, in-process reviews, to coordinate this cycle's activities with other cycles.

***Measurables***

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.12 COMMIT TO PLAN



This activity begins in Step 3, Plan Development.

### **Overview**

The Commit to Plan activity is an opportunity for stakeholders to review and comment on the results of the cycle planning and scheduling activities. Consensus that the activities of the cycle plan are appropriate to meeting cycle objectives is reached. The cycle plan may be updated according to stakeholders' comments.

When commitment to the detailed cycle plan is secured, a formal briefing to senior management on the direction the current cycle development activities will take can occur. This briefing may be a natural extension of any existing, periodic internal management reviews or may need to be scheduled separately. As a result of committing to the cycle plan, work packages are opened and assigned and the detailed cycle process, as documented in the cycle plan, is considered enactable.

### **Performers**

Key stakeholders, such as the project manager, risk analyst, technical lead, and engineer and customer

### **Inputs**

The input to this activity is the draft cycle plan.

### **Outputs**

- Meeting minutes
- The approved cycle plan

### **Entrance Criteria**

The entrance criterion for this activity is a draft cycle plan instantiated to an enactable level of detail.

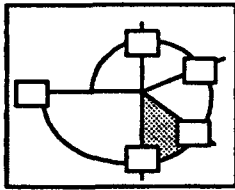
### **Exit Criteria**

- Consensus is reached regarding any changes to the draft cycle plan.
- All agreements or changes in the meeting minutes are documented and the minutes distributed to all attendees.
- Consensus is reached to proceed with the cycle by enacting the cycle plan.
- Work packages are opened.

### **Measurables**

- The total activity labor effort (see Section A.1)
- The number of changes to the cycle plan

### A.13 DEVELOP AND VERIFY PRODUCT



#### *Overview*

This activity begins in Step 4, Develop Product.

The product or part of the product necessary is developed per the cycle objectives, results are inspected for technical merit, and results are verified against cycle objectives and success criteria. Specific product development activities detailed in the cycle plan are enacted. That is, mainline or traditional software development activities are enacted as scheduled for the current cycle, such as requirements, design, code, integration, and test.

Verification is performed as an integral part of the fourth step development activities. Verification ensures that the artifacts or advanced product maturity produced by the development activities meet specified technical product and quality assurance requirements, cycle objectives, and success criteria. Verification techniques may be informal walkthroughs or reviews or formal inspections.

#### *Performers*

User, project manager, risk analyst, technical lead, engineer, and customer

#### *Inputs*

- The approved cycle plan
- Product baselines produced in previous cycles, if applicable

#### *Outputs*

- Product and cycle measurements and status
- Product or part of the product necessary to meet cycle objectives and satisfy cycle success criteria
- The enacted cycle plan

#### *Entrance Criteria*

The entrance criterion for this activity is an approved cycle plan instantiated to an enactable level of detail for the cycle.

#### *Exit Criteria*

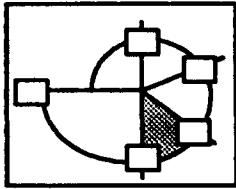
- The cycle process has been enacted as defined in the cycle plan.
- The product or part of the product necessary to meet cycle objectives and satisfy cycle success criteria has been built.
- Product components have been verified to satisfy all allocated requirements.
- Product components are verified as correct and consistent with the design, if applicable.
- Product components have been verified as maintainable and understandable.

***Measurables***

- Product components have been verified to comply with project standards.
- Actual total labor effort for each planned and scheduled cycle activity (see Section A.1)
- Actual number and type of product components (e.g., modules, documentation) produced
- Actual size of each product component
- Actual number, severity, and source of errors found during verification activities
- Product change or revision data, if any
- Actual risk increase or decrease of each high-priority risk
- Overall spiral risk

Refer to the *Software Measurement Guidebook* (Software Productivity Consortium 1992b) for more specific guidance on defining and collecting size, cost, schedule, and error measurements.

## A.14 MONITOR AND REVIEW



### Overview

This activity begins in Step 4, Develop Product.

The Monitor and Review activity maintains management control over the development process. This activity is performed in parallel with the Develop and Verify Product activity and periodically captures and analyzes the status of the cycle to provide management with insight into the cost, schedule, and technical performance status of the cycle.

This activity takes the raw activity progress status and analyzes it to produce management metrics that support decision making regarding corrective action. Collect and analyze the work package status in each cost account and summarize the status to the highest schedule level. For each parameter selected for monitoring, update the actual and projected performance, analyze the trends, and note unfavorable trends or values. The following should be reviewed and analyzed based on the data and measurements collected during the Develop and Verify Product activity:

- Product component size status
- Product component error data
- Product component change data
- Cycle schedule status
- Cycle cost status
- Cycle risk status
- Cycle quality assurance status
- Cycle organizational status

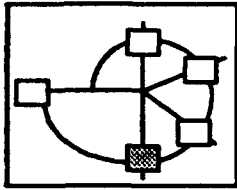
Refer to the *Software Measurement Guidebook* (Software Productivity Consortium 1992b) for specific size, cost, schedule, and error measurement and analysis methods.

Periodic cycle reviews are conducted as a part of this activity. These reviews serve two purposes. First, they demonstrate to the team and other stakeholders that the current cycle and overall spiral are under control. Second, they provide a mechanism to obtain a commitment to proceed with the cycle plan. At these reviews, decisions can be made to reallocate resources, replan schedules, or reassess risks for the current cycle activities. All such changes are reflected in an updated cycle plan/evolved cycle process.



<b><i>Performers</i></b>	Project manager, risk analyst, technical lead, and engineer
<b><i>Inputs</i></b>	<ul style="list-style-type: none"><li>• Product component and cycle measurements and status</li><li>• The approved cycle plan</li></ul>
<b><i>Outputs</i></b>	<ul style="list-style-type: none"><li>• Cycle status reports</li><li>• The updated cycle plan</li></ul>
<b><i>Entrance Criteria</i></b>	The entrance criterion for this activity is that the cycle process is being enacted as defined in the cycle plan.
<b><i>Exit Criteria</i></b>	<ul style="list-style-type: none"><li>• Product component and cycle data has been collected and reviewed.</li><li>• Status reports have been produced and distributed.</li></ul>
<b><i>Measurables</i></b>	The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.15 REVIEW TECHNICAL PRODUCT



This activity begins in Step 4, Develop Product.

### **Overview**

The technical product review is a stakeholder review of the product or part of the product developed to ensure that cycle objectives and success criteria were met. This review is an opportunity for stakeholder review; however, the ongoing Monitor and Review activity should have identified and resolved any factor that might have had a negative impact on meeting cycle objectives or success criteria.

### **Performers**

Key stakeholders, such as project manager, technical lead, engineer, and customer

### **Inputs**

The inputs to this activity are:

- The product or part of the product produced as a result of enacting the cycle plan
- The enacted cycle plan
- Spiral planning documents (see Section A.1)

### **Outputs**

- A product or part of the product that meets cycle objectives, success criteria, and quality assurance requirements and is approved for placement under configuration control.
- A configuration identification has been received.

### **Entrance Criteria**

The entrance criterion for this activity is that the cycle process as defined in the cycle plan has been enacted and has produced a product or part of the product.

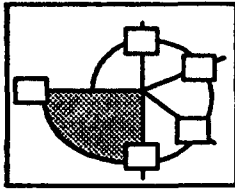
### **Exit Criteria**

- The product or part of the product has been reviewed to ensure that cycle objectives and success criteria were met.
- The results of verification activities have been reviewed to ensure the technical quality of the product or part of the product produced.
- Unique configuration identification has been assigned to the product or part of the product approved for placement under configuration control.

### **Measurables**

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.16 PRODUCT CHANGE CONTROL



This activity begins in Step 5, Manage and Plan.

### **Overview**

After cycle objectives and success criteria are met and the product developed in Step 4 is approved by the team during the technical product review, the product or part of the product produced during the cycle is baselined. Track the implementation of changes to controlled software products to ensure that the configuration of the product is evident at all times. Establish each baseline and track all subsequent changes in the configuration status. Maintain the history of changes to each configuration item throughout the software life cycle for status accounting. Store all items under control in a secure, limited-access software development library.

### **Performers**

Configuration manager

### **Inputs**

- The product or part of the product that meets cycle objectives, success criteria, and quality assurance requirements and has been approved for placement under product change control
- Configuration identification
- Spiral planning documents (see Section A.)

### **Outputs**

The output of this activity is a product baseline.

### **Entrance Criteria**

- A configuration management plan is included as one of the spiral planning documents.
- A product or part of the product has been developed as a result of enacting the cycle process and needs to place it under configuration control.
- A configuration identification has been received.

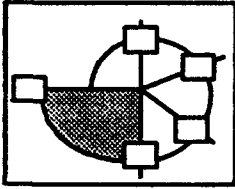
### **Exit Criteria**

The exit criterion for this activity is that baseline for the product or part of the product baseline has been established .

### **Measurables**

The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.17 REVIEW PROGRESS



This activity begins in Step 5, Manage and Plan.

### **Overview**

In the Review Progress activity, actual measures are evaluated against those estimated in the cycle plan, success criteria are examined to ensure that they were met, and lessons learned are identified. Analyze how the final cycle status may affect the process for the rest of the spiral. Specifically, review cycle data and verify the actual measurements described in the measurables section.

### **Performers**

Technical lead, engineer, contract manager, quality manager, project manager, and risk analyst

### **Inputs**

- Product baselines
- Actual product component and cycle measurements and data
- Cycle status reports
- Supporting documents (see Section A.1)
- Spiral planning documents (see Section A.1)

### **Outputs**

- Cycle and spiral metrics
- Updated spiral process drivers

### **Entrance Criteria**

- The cycle process as defined in the cycle plan has been enacted.
- A baselined product exists.

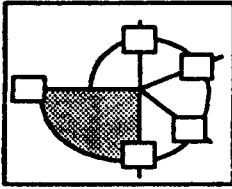
### **Exit Criteria**

- Final cycle status has been reviewed.
- Actual-versus-estimate metrics for both cycle and spiral have been produced.
- Lessons learned from the cycle have been compiled.
- Changes to spiral process drivers have been identified based on the enacted cycle process and resulting cycle product, data, and status.

### **Measurables**

- Number and type of product components
- Component size
- Activity, cycle, and overall spiral schedule
- Activity, cycle, and overall spiral labor hours and dollars
- Component and product errors
- The total activity labor effort (see Section A.1)

## A.18 UPDATE SPIRAL PLANNING DOCUMENTS



**Overview**

This activity begins in Step 5, Manage and Plan.

In the Update Spiral Planning Documents activity, the remainder of the spiral process as defined in the spiral planning documents is evolved to reflect enacted cycle process and metrics. Analyze the lessons learned, spiral and cycle status reports and metrics, and updated process drivers. Update the spiral process with the following information:

- Updated engineering and management procedures
- High-level schedule for remainder of the spiral
- Cost in dollars and labor hours for the remainder of the spiral
- Number of key spiral risks remaining
- Risk increase or decrease of each high-priority risk and overall spiral risk
- Errors identified, solved, and remaining
- Organizational status
- Lessons learned

Although much of this information may have been gathered and analyzed in previous activities, this activity gives the opportunity to examine thoroughly the impact of each part of the enacted cycle process on the rest of the product development process. For example, compose and analyze the answers to questions, such as:

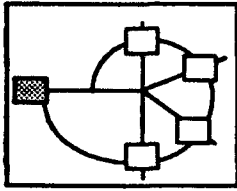
- How do the lessons learned affect the spiral schedule?
- How do the updated engineering and management procedures affect the increase or reduction of each high-priority risk?
- Should the labor hour estimates for the remainder of the spiral be updated as a result of the lessons learned?

Update the spiral planning documents to the level of detail appropriate to the current level of understanding of the spiral and its process drivers. That is, plan the next cycle activities to estimate the situation, analyze and avert risks, and plan development.

Refer to the *Software Measurement Guidebook* (Software Productivity Consortium 1992b) for specific size, cost, schedule, and error measurement and analysis methods.

<b><i>Performers</i></b>	Project manager, risk analyst, technical lead, corporate manager, contract manager, and quality manager
<b><i>Inputs</i></b>	<ul style="list-style-type: none"><li>• Updated spiral metrics and process drivers</li><li>• Supporting documents (see Section A.1)</li><li>• Spiral planning documents (see Section A.1)</li></ul>
<b><i>Outputs</i></b>	The output of this activity is the updated spiral planning documents.
<b><i>Entrance Criteria</i></b>	<ul style="list-style-type: none"><li>• All other activities in the current cycle have been completed.</li><li>• The enacted cycle plan has been included as one of the spiral planning documents and captures the following information:<ul style="list-style-type: none"><li>– Cycle metrics</li><li>– Cycle status reports</li><li>– Cycle lessons learned</li></ul></li></ul>
<b><i>Exit Criteria</i></b>	The exit criterion for this activity is that the spiral planning documents for the spiral have been updated.
<b><i>Measurables</i></b>	The measurement to be taken for this activity is the total activity labor effort (see Section A.1).

## A.19 COMMIT TO PROCEED



This activity begins in Step 5, Manage and Plan.

### **Overview**

The commit concept, which comes into play at the end of Step 5, is one of the most important in using the spiral model. It is similar in purpose to a baseline. All stakeholders should be briefed on the results of the current cycle and on any changes in plans and should agree with the decisions made regarding what to do next. Stakeholders include project management at all levels as well as the team. The review may also include customer representatives. The purpose of the review is to determine whether spiral-level objectives, alternatives, and constraints are still feasible and on track, agree on the objectives and success criteria for the next cycle, and commit resources to that cycle.

The spiral planning documents may be modified as the stakeholders reach agreement. Document all changes with the change rationale in the meeting minutes and distribute the minutes to all attendees.

### **Performers**

All project stakeholders, such as the technical lead, engineer, contract manager, quality manager, user, project manager, risk analyst, and customer

### **Inputs**

The input to this activity is the updated spiral planning documents.

### **Outputs**

- Meeting minutes
- Approved spiral planning documents

### **Entrance Criteria**

- The spiral planning documents are drafted for the spiral or updated following an enacted cycle.
- All individuals with an interest in the success of the spiral or their representatives are participating.

### **Exit Criteria**

- Consensus to proceed has been reached as a result of reviewing the spiral planning documents.
- Consensus on the objectives and success criteria for the next cycle has been reached.
- All agreements or changes to the spiral planning documents have been documented in the meeting minutes and the minutes have been distributed to all attendees.

### **Measurables**

- The activity labor effort (see Section A.1)
- The number and type of changes to the spiral planning documents

## **APPENDIX B. PRODUCT DEVELOPMENT ACTIVITY SPECIFICATIONS**

This appendix identifies and specifies a set of typical product development activities. The specifications do not state how to perform the activities; they are designed to be tailored based upon key organizational, business area, and/or project characteristics. Some characteristics to consider include organizational policies and procedures, acquisition strategy, project size and complexity, system requirements, and development methods. Tailoring may include the deletion of nonapplicable activities, modification of activities to include additional or more stringent requirements, multiple instances of the same activity as needed, or addition of other activities.

No temporal order is imposed on the activities. If sufficient inputs are available and the entrance criteria are satisfied, an activity can begin. A partial ordering among the activities is implied, however, in that the output of one activity may be required as the input to another activity. The performance of an activity is complete when all of its required tasks are performed in accordance with the exit criteria. This may require several iterations of the activity.

### **B.1 SOFTWARE SYSTEMS ENGINEERING**

The Software Systems Engineering activity class includes all the activities related to a system's specification, design, and integration.

#### **B.1.1 Specify Operational Concept**

##### ***Overview***

In the Specify Operational Concept activity, you define the system mission and concept of operations. Your primary goals are to:

- Define a system mission that clarifies the true reason for the system and ensures that the right system will be built.
- Define an operational concept for the system that meets mission requirements and external user and system operations needs.

The operational concept provides the initial top-level description of the system. It assesses the end-to-end system scope and the feasibility of proceeding with the development effort. This includes identification of potential risks and potential high-cost areas.

Within this activity, you:

- Identify the initial description of system capability and elicit ideas on alternative system concepts by analyzing the input that initially identified the desired operational capabilities.



	<ul style="list-style-type: none"><li>• Identify critical technological limitations, costs drivers, and risks that may affect system development.</li><li>• Assess the technical, organizational, political, legal, and economic feasibility of both development and operations.</li></ul>
<b>Performers</b>	User, systems engineer, project manager, customer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Customer/user requests</li><li>• Market data</li><li>• Technical support requests</li><li>• Operational capabilities</li><li>• Company ideas</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Operational concept</li><li>• System capability</li></ul>
<b>Entrance Criteria</b>	<p>The entrance criterion for this activity is that you have received information from one or more of the following input sources, indicating desired operational capabilities for a new or existing system:</p> <ul style="list-style-type: none"><li>• Market data</li><li>• Customer/user requests</li><li>• Internal ideas</li><li>• Requests from operation and maintenance activities</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have baselined an operational concept that is feasible, consistent, and complete with respect to the input source(s).</li><li>• You have defined an initial description of system capability describing what the system will do.</li></ul>
<b>Measurables</b>	The measurement to be taken for this activity is the total labor effort for this activity.

### **B.1.2 Formulate Potential Approaches**

#### **Overview**

In the Formulate Potential Approaches activity, you need to formulate a set of potential approaches based on input from the development environment, project risks, the target environment, a system capability, and the operational concept. You may include such different approaches as buying the system, developing a new system, or modifying an existing system.

Within this activity, you:

- Analyze the system capability to understand the system and identify those capability areas that are directly affected by different development approaches.
- Develop an understanding of the technology that may be needed to develop the system, including the use of commercial or third-party tools.
- Explore and analyze several different approaches for technical, organizational, political, legal, and economic feasibility.
- Analyze critical and high-risk technical and operational concepts.

For each approach, you need to include a description of the constraints, benefits, and risks. From the list of potential approaches, you identify any recommended approach(es) with the associated justification. You should discard only those approaches that do not work and document all other approaches in the recommended approaches artifact. The Recommend Software System Development Strategy and the Plan Development activities will use this artifact with other project information to determine the specific approach to take.

#### **Performers**

Systems engineer, project manager, risk analyst

#### **Inputs**

- Operational concept
- Target environment
- RMP
- System capability

#### **Outputs**

The output from this activity is the recommended approach.

#### **Entrance Criteria**

- You have received a baselined operational concept.
- You have received a defined system capability.
- You have identified the target environment.
- A risk assessment is available that identifies known project risks.

***Exit Criteria***

The exit criterion for this activity is that you have selected and baselined a recommended approach along with an associated justification.

***Measurables***

- The total labor effort for this activity
- The number of approaches generated in relation to the size of the system

### **B.1.3 Define System Requirements**

#### ***Overview***

In the Define System Requirements activity, you collect, integrate, specify, relate, and organize the customer's needs and objectives to provide the foundation for system design and implementation. As your input, you use:

- The operational concept or approved product change data
- An assessment of the current, known project risks
- The requirements analysis results
- A software system development strategy that identifies any constraints on the requirements due to how the system is going to be developed
- Any constraints defined in the cycle development plan

Within this activity, you:

- Collect system requirements information, including mission-driven milestones, from the customer and users.
- Organize the requirements into categories to aid in checking completeness and in determining the accuracy of the requirements.
- Quantify the requirements into measurable entities whenever possible.
- Analyze system requirements for completeness, consistency, testability, validity, verifiability, and feasibility.

You use the known project risks to help identify those requirements that are high risk, for example, requirements that are less stable, require the use of new technology, or require unavailable resources. You then take appropriate risk mitigation strategies, such as addressing the high-risk requirements first and/or isolating them from other requirements. Another approach to reduce risk is to request an analysis of the system requirements for specific problem areas in the hope of better understanding any problems. You request that analysis activities be performed to clarify the risk in high-risk requirements.

You develop a verifiability checklist for each requirement that you can use to verify each requirement. You also develop a requirements flow diagram that you use to verify the data structure or functional behavior of each artifact in the system and to ensure that it satisfies its supporting requirement.

You may define only part of the system requirements with a single pass through this activity. If this is the case, then you must reenter this activity to complete the requirements.

<b><i>Performers</i></b>	Systems engineer, project manager, user, customer
<b><i>Inputs</i></b>	<ul style="list-style-type: none"><li>• Operational concept</li><li>• Cycle plan</li><li>• RMP</li><li>• System analysis results</li><li>• Software system development strategy</li><li>• System capability</li><li>• Recommended approach</li><li>• Product change data</li></ul>
<b><i>Outputs</i></b>	<ul style="list-style-type: none"><li>• System requirements</li><li>• Analysis requests</li><li>• Verifiability checklists</li><li>• Requirements flow diagram</li></ul>
<b><i>Entrance Criteria</i></b>	<ul style="list-style-type: none"><li>• There is a need to define or modify the system requirements based on one or more of the following inputs:<ul style="list-style-type: none"><li>– A baselined operational concept</li><li>– Approved product change data that defines changes to the system requirements</li><li>– Results of system requirements analysis activities</li><li>– Results of identifying the software system development strategy</li></ul></li><li>• You have received a defined system capability.</li><li>• You have received a baselined recommended approach.</li><li>• A risk assessment is available describing known project risks.</li><li>• You have received a baselined cycle development plan.</li></ul>
<b><i>Exit Criteria</i></b>	<ul style="list-style-type: none"><li>• You have baselined the system requirements that you defined and/or modified during an iteration through this activity and ensured that they satisfy the following criteria:<ul style="list-style-type: none"><li>– They satisfy customer needs.</li><li>– They comply with system constraints.</li><li>– They are consistent with each other.</li></ul></li></ul>

- They are verifiable.
- They are feasible.
- They are understandable.
- You have baselined the verifiability checklists and the requirements flow diagram.
- You have documented any requests to analyze the system requirements for specific characteristics.

***Measurables***

- The number of requirements
- The number of requirements added, deleted, or modified during an iteration through this activity
- The total labor effort for this activity in comparison to the size of the system

### **B.1.4 Develop System Architecture**

#### ***Overview***

In the Develop System Architecture activity, you identify a system-level architecture, including hardware and software components, that satisfies both initial and future requirements within budget and schedule constraints. Within this activity, you:

- Transform the operational concept and the system requirements or an Engineering Change Proposal (ECP) into a system architecture based on the recommended approach.
- Derive the specific configuration of hardware, software, and firmware to perform required functions at the necessary level of performance and dependability.
- Analyze the technical suitability and cost of the architecture.

You need to make modifications to the recommended approach, verifiability checklists, or requirements flow diagram based on the results of developing the system architecture and to rebaseline the artifact. You also need to document changes to the software requirements based on the results of developing the system architecture.

To mitigate risks associated with the system architecture, you can develop and analyze multiple system architectures before you select a preferred architecture for development. You should base your analysis of the candidate architectures on both technical and cost considerations. Technical considerations include:

- The ability to handle expected workloads
- The ability to handle expected operational scenarios
- Both the environment in which you will develop the system and the environment in which you will deliver it
- The ability to support the software system development strategy
- Commonality and reuse
- People as components of the system, describing their behavior and needed system functionality
- System reliability, maintainability, and availability
- System safety
- Human factors
- The user interface design
- System security

This guidebook details the activities for the analysis of several key architecture considerations. Cost considerations include software cost drivers, such as size and reuse, and hardware cost drivers, such as system capacity, system reliability and availability, hardware maintenance, and environmental considerations. The cycle development plan provides the constraints and methods that you need to follow when performing the activity. You may possibly develop the system user interface design concurrently with this activity.

You should use the risk assessment to help identify those architecture components that have the highest risk, for example, components that require new technology or have a high complexity level. You should then take the appropriate risk mitigation strategies, such as addressing these components first to minimize impact on the life cycle. You request to have an analysis of specific characteristics performed on high-risk components to help understand any problems.

You should address several system considerations when deriving the top architectural level to offset larger problems or risks that may occur later. These system considerations include:

- ***Allocation of Functionality.*** Due to budget or resource constraints, you may need to limit the initial functionality of the system to fall within a subset of the users' requirements. You may add additional functionality at a later date.
- ***System Automation.*** Varying degrees of system automation are available, and you must determine them based on the needs of the system and the desires of the users.
- ***Stable Versus Volatile Requirements.*** You must handle volatile requirements differently from stable requirements within the system architecture. You must directly accommodate the volatile requirements in the system architecture.
- ***Modular System Design.*** Modular designs ease planned evolution, implementation, and enhancement. You may use an "open backbone" architecture to address the end-to-end system requirements while permitting the addition of future functionality. However, you need to make tradeoffs concerning the number and size of the architecture components. Small components are easier to implement and maintain, but a large number of components complicates maintenance and logistics issues.
- ***Built-In System Fault Detection, Isolation, and Recovery.*** Systems with stringent maintainability and availability requirements require this feature. In addition, you might have to support a built-in system backup. These features might require special hardware, software, or firmware requirements.



- ***Expandability and Flexibility.*** You should determine where you anticipate growth and change and see whether you can accommodate them within the architecture. You should define these determinations as system requirements.
- ***Commonality.*** You should analyze the architecture for the potential use of common system elements. The multiple use of these elements across architectural components can simplify design and implementation.

Within this activity, you perform the major tasks of decomposing the system requirements into lower level requirements and allocating the requirements to the architectural components to ensure that you address each system requirement. Both of these major tasks are described in more detail below.

- ***Decompose System Requirements.*** Based on the analysis of the system requirements, you further decompose the system requirements into lower level requirements so that you can implement them in a single hardware configuration item (HWCI) or computer software configuration item (CSCI). You should structure the requirements to determine their interdependency and relative priority.
- ***Allocate Requirements to Hardware, Software, and Interface Components.*** You allocate the system requirements to the architectural components to ensure that you have adequately addressed each requirement. Some rules that you can follow in allocating the requirements are:
  - You should group together those requirements that are tightly coupled to minimize communications.
  - For large systems, you should first allocate only those requirements that affect the architecture, such as requirements that have a heavy workload, storage, or user interface requirements.
  - You should investigate each requirement and develop alternative allocations to system architectural components for implementation, for example, to software, firmware, or a special purpose processor.
  - You should realize that some functions, such as screen editing or network controls, may be able to migrate between architectural components.
  - You should ensure that the system user interface design is allocated to the appropriate architectural components.

***Performers***

Systems engineer, software engineer

***Inputs***

- Recommended approach
- Cycle plan

- System requirements
- RMP
- System analysis results
- Software system development strategy
- Software engineering environment
- Target environment
- Verifiability checklists
- Requirements flow diagram
- Product change data
- System user interface design

***Outputs***

- Recommended approach
- System architecture
- Analysis requests
- Verifiability checklists
- Requirements flow diagram
- Product change data

***Entrance Criteria***

- There is a need to develop or modify the system architecture based on one or more of the following inputs:
  - Baselined system requirements
  - Approved product change data defining changes to the system architecture
  - Results of system architecture analysis activities
  - Results of identifying a software system development strategy
- You have received a baselined operational concept.
- You have received a baselined recommended approach for developing the system.
- You have identified both the software engineering environment and the target environment.

- You have received a baselined cycle development plan.
- You have received a risk assessment documenting all known project risks.
- You have received baselined verifiability checklists and a baselined requirements flow diagram.
- Any part of the system user interface design that has been completed is a variable.

***Exit Criteria***

- You have baselined the system architecture defined/modified during an iteration through this activity and ensured that it meets the following criteria:
  - It is traceable to the system requirements.
  - It is consistent with the system requirements.
  - It addresses each system requirement.
  - It is compliant with project standards.
  - It is feasible.
  - It is maintainable.
- You have decomposed the system requirements to a level where they can be addressed by a single hardware, software, or interface module.
- You have baselined any changes to the recommended approach.
- You have documented any requests for analysis of specific architecture features.
- You have documented any recommendations to change the software requirements in the product change data.
- You have made and baselined any changes to the verifiability checklists or requirements flow diagram.

***Measurables***

- The total labor hours allocated to this activity in relation to the size of the system architecture
- The number of architectural components and number of interfaces
- The number of architectural components and number of interfaces added, deleted, or modified during an iteration through this activity

### B.1.5 Design User Interface

#### **Overview**

You design the user interface primarily by assessing different user interfaces for how they address several important characteristics. You need to take into consideration the human factors requirements that you defined with the customer and the user, any part of the system architecture that you have developed, and the target environment under which the system will operate. The cycle development plan describes constraints and methods under which to perform the activity; for example, the cycle development plan describes standards to follow when designing the interface. In addition, you need to take into consideration any specific requests from the system requirements or system architecture development teams to analyze specific usability features and incorporate the results of the analysis into the user interface design. You may develop the user interface concurrently with the system architecture.

You should find a design that has acceptable user interface characteristics and demonstrate that design to the customer and user to receive agreement that the interface meets the customer's requirements. Based on the results of the design, you should make and document recommendations to modify the system requirements and/or the system architecture.

Usability is the measure of how well people will be able and motivated to use the system practically (Gilb 1988). Therefore, in designing the user interface, you should address:

- ***User Characteristics.*** The interface should be consistent in terms of what knowledge level the user must have to use the system practically, or it should make clear any deviations from the norm.
- ***Ease of Use.*** The interface should encourage a low level of user learning time; minimize the need for training, manuals, and consulting; and maintain a high level of user satisfaction.
- ***User Control.*** The interface should allow the user to take control while assisting him in performing his functions. The interface should avoid automatic actions or actions requiring heavy computation that require no user interaction or choice. The goal of this characteristic is to increase user productivity.
- ***Invalid Input Handling.*** The interface should provide the user with a high probability of entering correct data. It should provide prompt and easy detection and recovery of invalid input.
- ***Reliability and Robustness.*** The interface should have a minimum number of possible malfunctions and should handle system malfunctions by protecting the user.
- ***Consistency.*** The interface should have a consistent "look and feel" across displays, interactions, user input, and system feedback.

- **Transparency.** The interface should allow the user to perform the tasks without being aware of the lower level system mechanics.
- **Efficiency.** The interface should minimize user effort so that performing a function with the system is easier than performing a function without the system.
- **Adaptability.** The interface should adapt to the different types of users; that is, it should accommodate one user wishing to work through a menu interface and another user wishing to work through a command line interface.
- **Expandability.** The interface should be flexible enough to accept modifications to existing components as well as the addition of new components.

You should make the user interface design that is selected for and demonstrated to the customer available to the software engineers for implementation support. You will allocate the user interface design to the system architecture components in the Develop System Architecture activity.

**Performers**

Systems engineer, user

**Inputs**

- Human factors requirement
- Cycle plan
- System architecture
- Target environment
- Analysis requests
- Product change data

**Outputs**

- System user interface design
- Product change data

**Entrance Criteria**

There is a need to design or modify the user interface based on one or more of the following inputs:

- You have received baselined human factors requirements.
- You have approved product change data that defines changes to the user interface design.
- You have identified the target environment.
- You have received an initial system architecture.
- You have received a baselined cycle development plan.

***Exit Criteria***

- The user interface is acceptable to the customer.
- You have documented any changes to the system requirements or the system architecture to support a better user interface in the product change data.

***Measurables***

The measurement to be taken for this activity is the total labor effort in relation to the size of the user interface component of the system.

### **B.1.6 Analyze System Performance**

**Overview**

In the Analyze System Performance activity, you analyze the system architecture to ensure conformance to system performance requirements or to satisfy a request for analysis against specific performance features. The performance analysis takes into consideration the target environment in which the system will operate and any constraints and methods identified in the cycle development plan.

Standard performance features that you should analyze include:

- Capacity analysis or the system's ability to ensure sufficient memory, communication throughput, speed, and response time performance while allowing for growth and reserves
- Workload or the major system drivers on resource components such as computer throughput, memory, and input/output bandwidth
- Workflow or the workload of the subsystem architecture determined by the transactions and transaction rates against the capacity levels

You record the results of your analysis along with recommendations on how the system or software engineer should develop the system or how the system requirements or system architecture should be modified to maximize performance.

**Performers**

Systems engineer, risk analyst

**Inputs**

- Analysis requests
- System architecture
- Performance requirement
- Target environment
- RMP
- Cycle plan

**Outputs**

- System performance analysis results
- Product change data

**Entrance Criteria**

There is a need to analyze the performance of the system architecture based on one or more of the following inputs:

- You have received the baselined performance requirements.
- You have received a request to analyze the system for specific performance features.

- You have received a risk assessment identifying system performance as a high risk.
- You have received a system architecture defined to a level detailed enough to allow for analysis of performance characteristics.
- You have identified the target environment.
- You have received a baselined cycle development plan.

***Exit Criteria***

- You have documented and completed the results of the system performance analysis and proposed them to the customer and user for acceptance.
- You have documented any changes to the system architecture or system requirements that will improve performance in the product change data.

***Measurables***

- The estimated versus actual performance of certain system configurations
- The number and severity of changes that this activity generates to the system requirements and/or system architecture



### **B.1.7 Analyze System Dependability**

#### ***Overview***

You need to analyze the system architecture for conformance to system dependability requirements or to satisfy a request for analysis of specific dependability features. The analysis should take into consideration both the target environment under which the system will operate and the constraints and methods identified in the cycle development plan.

System dependability requirements that you may analyze include:

- ***Reliability.*** These issues include inherent failure distributions and failure rates for the hardware and the number of errors detected and their rate of detection for the software.
- ***Maintainability.*** These issues include the average amount of time required to repair a failed entity and to restore normal operation.
- ***Availability.*** These issues include the fraction of total system operating time during which the intended function of the system is being fulfilled.
- ***Security.*** These issues include whether someone can infiltrate the system by bypassing the security measures and facilities designed to prevent such actions.
- ***Survivability.*** These issues include the system's ability to perform and support critical functions without failures within a specified time period when a portion of the system is inoperable.
- ***Safety.*** This is a measure of the time to catastrophic failure from a referenced initial instant.

You should compare the results of the analysis of these dependability issues to the dependability requirement, and should document any recommended changes to the system requirements or system architecture to address any deviations from the requirements.

#### ***Performers***

Systems engineer, risk analyst

#### ***Inputs***

- Analysis requests
- Dependability requirement
- System architecture
- Cycle plan
- Target environment
- RMP

<b><i>Outputs</i></b>	<ul style="list-style-type: none"><li>• System dependability analysis results</li><li>• Product change data</li></ul>
<b><i>Entrance Criteria</i></b>	<p>There is a need to analyze the system architecture for dependability based on one or more of the following inputs:</p> <ul style="list-style-type: none"><li>• You have received the baselined system dependability requirements.</li><li>• You have received a request to analyze the system architecture for specific dependability features.</li><li>• The risk assessment identifies system dependability as a high risk.</li><li>• You have received a system architecture defined to a level detailed enough to allow for analysis of dependability characteristics.</li><li>• You have identified the target environment.</li><li>• You have received a baselined cycle development plan.</li></ul>
<b><i>Exit Criteria</i></b>	<ul style="list-style-type: none"><li>• You have documented and completed the results of the system dependability analysis and proposed them to the customer and user for acceptance.</li><li>• You have documented any changes to the system architecture or system requirements that will improve system dependability in the product change data.</li></ul>
<b><i>Measurables</i></b>	<ul style="list-style-type: none"><li>• The relative dependability of certain system configurations</li><li>• The estimated versus actual dependability of certain system configurations</li></ul>

### **B.1.8 Analyze System Reusability**

#### ***Overview***

You assess the system architecture and/or the system requirements for the feasibility of incorporating reusable components by using the following steps:

- Identify opportunities for reuse.
- Identify a reusable part that is a close or perfect match to some subset of the system.
- Evaluate that part to determine any modifications that you need to make to either the part, the system requirements, or the system architecture.
- Determine whether it will be more beneficial to include the part and make any necessary modifications or whether it will be more beneficial to not use the part and develop the system subset within the project.

This analysis takes into consideration the software engineering environment under which the system is being developed, the constraints and methods provided by the cycle development plan, and any reuse library filled with reusable components already established by the organization. You also analyze the architecture and the requirements to determine if any component of either artifact engineers can reuse in later systems or if slight modifications to the architecture or the requirements can make it more reusable.

You record the results of this analysis, identifying what reusable component(s) you recommend to be included in the system and describing how the software and systems engineers should develop the system to accommodate the component. You may also include recommendations for increased commonality within system architecture and/or system requirements to increase reusability across systems in the analysis results.

#### ***Performers***

Systems engineer, risk analyst

#### ***Inputs***

- System architecture
- Analysis requests
- Cycle development plan
- System requirements
- Reuse library
- Reusability requirement

#### ***Outputs***

- System reusability analysis results
- Product change data

***Entrance Criteria***

There is a need to analyze the system requirements and/or system architecture for reusability characteristics based on one or more of the following inputs:

- You have received baselined reusability requirements.
- You have received a request to identify reusable components that the software engineers can use to develop the system.
- You have received a request to analyze the system architecture or system requirements to identify which parts of the architecture or requirements the systems engineer can use as a reusable component.
- You have received the system architecture defined to a level detailed enough to analyze for reusability.
- You have received system requirements defined to a level detailed enough to analyze for reusability.
- You have baselined the cycle development plan.
- A reuse library is available.

***Exit Criteria***

- You have documented and completed the results of the system reusability analysis.
- You have documented any changes to the system architecture and/or system requirements that improve the reusability of the system in the product change data.
- You have identified and qualified all reusable components that the software and systems engineers will use to build the system.

***Measurables***

- The percent of the system requirements or system architecture that reusable components can partially or completely satisfy
- The number of new system requirements or architectural components versus the number of reused system requirements or architectural components
- The number of reused system requirements or architectural components that the systems engineer had to modify and the number that they did not have to modify

### **B.1.9 Analyze System Modifiability**

#### **Overview**

In the Analyze System Modifiability activity, you analyze the system architecture for its ability to be modified. Your analysis needs to take into consideration any constraints provided in the cycle development plan.

Modifiability is the measure of the system's ability to be improved, extended, and ported during its life cycle. Because you can change any system in almost any way given enough time, resources, and money, modifiability is a measure of a system's ability to change in an efficient manner.

Modifications to the system may be due to a change in users or market segment, purpose, environment (i.e., single versus multiuser, hardware versus software configuration), or user workload. Specific characteristics within a system architecture that would support such modifications include:

- **Improvability.** This is a measure of the efficiency of making major adaptations, changes, and improvements to the system. It differs from maintainability in that a fault is not currently present in the system; rather, you are improving a working system.
- **Extendability.** This is a measure of the ease of adding new factors to an existing system.
- **Portability.** This is a measure of the ease of moving a system from one environment to another (Gilb 1988).

You also measure a system's modifiability to ensure conformance to any system requirements requiring changes to the system after deployment; for example, you may have to port a system to another target environment after deployment or you may have to make technology upgrades to a long-lived system. You should record your findings of this analysis along with recommendations of how the software and systems engineers should develop the system or how the system architecture and/or system requirements the systems engineer should change to accommodate later system modifications.

#### **Performers**

Systems engineer, risk analyst

#### **Inputs**

- Cycle plan
- System architecture
- Analysis requests
- System requirements

#### **Outputs**

- System modifiability analysis results
- Product change data

***Entrance Criteria***

- You have received the baselined system requirements.
- You have received the baselined cycle development plan.
- You have received a system architecture defined to a level detailed enough to allow for analysis of modifiability characteristics.
- You have received a request to analyze the system architecture for certain modifiability features.

***Exit Criteria***

- You have documented and completed the results of the system modifiability analysis.
- You have documented the analysis results and your recommended changes to the system architecture or the system requirements to improve modifiability of the system into the product change data document.

***Measurables***

- The total labor effort for this activity.
- The total labor effort spent on fixing system problems enhancing the system after delivery, and porting the system to another platform in relation to the size of the system

### **B.1.10 Analyze System Functionality**

#### ***Overview***

In the Analyze System Functionality activity, you analyze the system architecture for specific functionality characteristics. Input to this activity includes the system requirements, the software engineering environment under which the software engineers will develop the system, and the constraints and methods provided in the cycle development plan.

The functionality characteristics you analyze should support the operational, test, support, distribution, marketing, and production requirements of the system. Specifically, functionality characteristics include how the architecture supports and addresses:

- The goals and requirements of the system
- The scope of planned system automation
- The quality of the decomposition of the architecture
- The range of applicability (e.g., does the architecture support multiusers and single users, does the architecture support different workloads, and does the architecture support different user hardware and software environments)
- Any standards or government regulations

Functionality characteristics also include how the system compares with any competition, a complexity analysis among the hardware and software components and their interfaces, and support for evolution

You should document any recommended changes to the system architecture or the system requirements to improve the system's functionality. You should also document the analysis results

#### ***Performers***

Systems engineer, risk analyst

#### ***Inputs***

- System requirements
- Cycle plan
- System architecture
- Analysis requests
- Software engineering environment

#### ***Outputs***

- System functionality analysis results
- Product change data

#### ***Entrance Criteria***

- You have received a system architecture defined to a level detailed enough to allow for analysis of functionality characteristics.

- There is a request to analyze the system architecture for functionality features.
- You have received the baselined system requirements.
- You have identified the software engineering environment.
- You have received the baselined cycle development plan.

***Exit Criteria***

- You have documented and completed the results of the system functionality analysis, and they are acceptable to the customer.
- You have documented any changes to the software requirements or the system architecture that will improve the functionality of the system in the product change document.

***Measurables***

The measurement to be taken for this activity is the total labor effort for this activity.



**B.1.11 Recommend Software System Development Strategy****Overview**

In the Recommend Software System Development Strategy activity, you define a strategy that provides the most appropriate approach to developing, integrating, and delivering the hardware and software systems. A major part of the strategy is a decision to use either a monolithic or an evolutionary development process. Decisions that you may make include putting off high-risk development areas to later deliveries or, if these areas represent a high expected value, including them as part of the initial development cycles.

Input to this activity comes from the system requirements, system architecture, the project's schedule (to allow you to understand delivery requirements), the cycle development plan (to allow you to understand any activity constraints or techniques that you need to follow), available resources, and the approach(es) selected for development. You perform this activity concurrently with the definition of the system requirements, system architecture, and project schedules because how the software and systems engineers are going to develop them has a direct impact on the cost of system development and how the software and systems engineers can develop the software system.

In this activity, you develop a strategy for developing and delivering a series of system builds to deliver increasing functionality to the customer during the development period. This reduces potential problems to manageable levels that the project, customer, and user can address with clear corrective actions by providing a checkpoint for project management and the customer to assess the technical, schedule, and cost status of a system development effort. The incremental implementation strategy should include a series of delivered system builds, each a combination of fully or partially completed configuration items and each adding increasing functionality to a proven baseline. The last build should encompass the entire system.

In this activity, you also develop a strategy for developing the system documentation that is consistent with the software system development plan.

**Performers**

Technical lead, systems engineer, project manager, documentation manager

**Inputs**

- Recommended approach
- System requirements
- Cycle plan
- Project master schedule
- System architecture
- Available resources

**Outputs**

- Software system development strategy
- System build definition

***Entrance Criteria***

- You have received system requirements defined to a level detailed enough to start identifying a software system development strategy.
- You have received a system architecture defined to a level detailed enough to start identifying a software system development strategy.
- You have received the project master schedule defined to a level detailed enough to start identifying a software system development strategy specifically identifying contract-specified deliverable dates.
- You have received a baselined cycle development plan.
- You have received a baselined recommended approach.
- You have identified the project resources, including environment, facilities, and personnel.

***Exit Criteria***

- You have developed a software system development strategy that includes the rationale for selecting the process model and the rationale for an incremental implementation schedule if you chose an incremental build or an evolutionary development-based process model.
- You have baselined a documentation development strategy that is feasible and consistent with the software system development strategy.
- You have baselined the system build definition.

***Measurables***

- The total labor effort
- The number of system builds scheduled versus the size of the system (number of architectural components)

### **B.1.12 Integrate Software System Components**

<b>Overview</b>	In the Integrate Software System Components activity, you integrate all CSCIs and HWCIs into a single system build according to the system build definition and the system architecture. This includes ensuring that the interfaces across the CSCIs and HWCIs are complete and that you have loaded each CSCI onto the appropriate HWCI. The cycle development plan provides any constraints and describes the method by which you should perform this activity.
<b>Performers</b>	Systems engineer, configuration manager
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Cycle plan</li><li>• HWCI</li><li>• System architecture</li><li>• CSCI</li><li>• System build definition</li></ul>
<b>Outputs</b>	The output from this activity is the system.
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• You have received a baselined system architecture.</li><li>• You have received a baselined system build definition.</li><li>• You have received all of the developed and verified CSCIs and HWCIs needed for this build.</li><li>• You have received a baselined cycle development plan.</li></ul>
<b>Exit Criteria</b>	The exit criterion for this activity is that you have integrated all needed CSCIs and HWCIs into a single system build.
<b>Measurables</b>	The measurement to be taken for this activity is the total labor effort.

## **B.2 SOFTWARE ENGINEERING**

The Software Engineering activity class includes all activities related to software requirements definition, design, implementation, and integration.

### **B.2.1 Define Software and Interface Requirements**

#### ***Overview***

In the Define Software and Interface Requirements activity, you analyze the system requirements to generate derived requirements that further clarify the allocated requirements and result in a complete set of CSCI engineering requirements, including the following:

- Functionality and capability specifications, including performance, quality, and physical characteristics and environmental conditions under which the software will perform
- Safety specifications, including those related to methods of operation and maintenance, environmental influences, and personnel injury
- Security specifications, including those related to compromising sensitive information or materials
- Human-engineering and man-machine specifications, including those related to manual operations, human-equipment interactions, constraints on personnel, areas needing concentrated human attention that are sensitive to human errors, and training
- Hardware processing resource and reserve specifications for processors, memory devices, and data channels
- Database requirements
- Installation and acceptance requirements of the delivered software at the operation and maintenance site(s)
- User operation and execution requirements
- User maintenance requirements
- Quality-oriented requirements
- Qualification requirements necessary to show that the software component complies with the established requirements
- Interface requirements that specify the protocols, priority, and concurrency of the software components

You use the risk assessment to identify incomplete, missing, or volatile requirements and to request that analysis activities, such as simulations, modeling, and prototyping, be performed to further clarify those

requirements. You also specify a set of product evaluation criteria that the software engineers will use during software system verification to ensure that the system meets all the software requirements.

You should iterate through this activity to progressively refine the software requirements. You negotiate changes to the system requirements if necessary and beneficial. You assess the impact of any proposed changes to the software requirements and implement any approved changes.

***Performers***

Customer, user, software engineer, systems engineer

***Inputs***

- System requirements
- Cycle plan
- RMP
- Software analysis data
- Software engineering environment
- Product change data

***Outputs***

- Software requirements
- Analysis requests
- Product change data

***Entrance Criteria***

- There is a need to define or modify the software requirements based on one or more of the following inputs:
  - A baselined set of system requirements
  - An approved change to the software requirements
- A risk assessment is available that identifies ambiguous, volatile, or conflicting system requirements.
- You have received the baselined cycle development plan.
- The software engineering environment is available to perform requirements analysis.

***Exit Criteria***

- The software requirements defined or modified during this iteration of the activity meet the following criteria:
  - They satisfy the customer's needs.
  - They are traceable to the system requirements.

- They comply with system and software constraints.
- They are consistent with each other.
- They are verifiable.
- They are feasible.
- They are understandable.
- You have issued requests to analyze any ambiguous, incomplete, or conflicting requirements.
- You have issued any proposed changes to the system requirements.
- The number of software and interface requirements that you derived, modified, or deleted during this iteration of the activity
- The number of changes that you proposed to the system requirements
- The number of labor hours required to define the software and interface requirements

***Measurables***

## B.2.2 Design Software Architecture

### *Overview*

The goal of the Design Software Architecture activity is to derive a software system configuration that performs the required functions at the necessary level of performance and reliability. The results of the design effort should be a software architecture representing an implementable, maintainable, and reusable product that meets all technical, cost, and schedule considerations. This activity consists of the following tasks:

- You transform the CSCI software requirements and the system architecture into a software architecture that describes the top-level structure, identifies major components, and indicates the hierarchy of control for each CSCI.
- You apply the CSCI requirements to the computer software components (CSCs) and further refine them to facilitate detailed design.
- You partition software requirements into derived requirements, or you generate new requirements to describe the component's purpose. You define the relationships, events, state behavior, and information domains of each CSC.
- When possible, you reuse software components and architectures from comparable systems.
- You use application standards, such as operating system interfaces, computer-human interfaces, and networking interfaces where appropriate.
- You should address the following considerations when deriving the software architecture:
  - *Allocation of Functionality.* Due to budget or resource constraints, you may need to limit the initial functionality of the software system to fall within a subset of the software requirements.
  - *Volatile Requirements.* You may wish to allocate unstable requirements to isolated architectural components.
  - *Modular Design.* Modular designs are more flexible and ease planned evolution, implementation, and enhancement.
  - *Commonality.* You should analyze the architecture for common software components. The multiple use of these components simplifies implementation and maintenance.

The risk assessment identifies potential high-risk areas in the design. If needed, you should perform analysis activities to obtain the additional engineering information needed to understand those portions of the design.

You develop and document a top-level design for the inter- and intra-CSCI interfaces. You define the message formats, calling parameters, and priorities of each interface.

You establish software build definitions that define the incremental development of functional subsets of the software.

***Performers***

Software engineer

***Inputs***

- System architecture
- Software requirements
- Cycle plan
- Software analysis data
- RMP
- Database description
- Software engineering environment
- Reuse library
- Product change data

***Outputs***

- Software system architecture
- Analysis requests
- Software build definition
- Product change data

***Entrance Criteria***

- There is a need to develop or modify the software architecture based on one or more of the following inputs:
  - A baselined set of software requirements
  - A baselined system architecture
  - A baselined database description
  - An approved change to the software architecture
- You have received the baselined cycle development plan.
- A risk assessment is available that identifies high-risk areas in the software system architecture.
- The software engineering environment is available to develop the software architecture.



***Exit Criteria***

- The software architectural design that you created or modified during this iteration of the activity meets the following criteria:
  - It is traceable to the software requirements.
  - It is consistent with the software requirements.
  - It is compliant with project standards.
  - It is feasible.
  - It is maintainable.
- You have specified the functionality to be included for each of the software builds leading to an integrated CSCI.
- You have issued requests to analyze any high-risk areas of the design.
- You have issued any proposed changes to the database description or software requirements.

***Measurables***

- The number of component designs you added, modified, or deleted during this iteration through the activity
- The number of changes you proposed to the software requirements or the database description
- The number of labor hours required to define the software system architecture

### **B.2.3 Analyze Software Performance**

**Overview**

In the Analyze Software Performance activity, you determine whether the software timing and sizing requirements are consistent and complete. You analyze the software system architecture to ensure that the architecture satisfies these critical requirements during system operation. You should analyze standard performance features, such as the amount of overhead the operating system uses to schedule each program, expansion factors for source code, effective communication transfer rates, and database throughput through simulation, modeling, or other appropriate means. Concentrate on software that is particularly critical to system performance, such as interrupt handlers, network protocols, etc.

You document the results of the analysis and any proposed changes to the software system architecture and the software requirements that will improve the performance of the software.

**Performers**

Software engineer, risk analyst

**Inputs**

- Software requirements
- Software system architecture
- System performance analysis results
- RMP
- Cycle plan
- Database description
- Analysis requests

**Outputs**

- Software performance analysis results
- Product change data

**Entrance Criteria**

- There is a need to analyze the performance of the software system architecture based on one or more of the following inputs:
  - The risk assessment identifies software performance as a high risk.
  - You have received a request to analyze the software for specific performance features.
  - The system performance analysis results indicate a problem with the software performance.
- You have received a baselined version of the software requirements.
- The database description is detailed enough to allow for the analysis of performance characteristics.

***Exit Criteria***

- You have received the baselined cycle development plan.
- You have documented and completed the results of the software performance analysis.
- You have proposed any changes to the software system architecture or to the software requirements that will improve performance to the Configuration Change Board (CCB).

***Measurables***

- The actual performance of certain software configurations compared to the performance estimates
- The number of changes you proposed to the software system architecture and/or the software requirements
- The total labor hours required to analyze software performance

## **B.2.4 Analyze Software Reusability**

<b>Overview</b>	<p>In the Analyze Software Reusability activity, you identify reusable components that the software engineer can use during software system development to satisfy part of the software requirements. You evaluate any modifications that need to be done to the component or the software architecture so that the software engineer can integrate it into the system. You then qualify the reusable component to verify that it is of high quality and satisfies all functional and nonfunctional requirements allocated to it.</p> <p>You identify any commercial off-the-shelf (COTS) software that the software engineer can use during software system development to satisfy part of the software requirements. You submit resource requests to project managers so that they can purchase the COTS software. You also analyze the software requirements and the software system architecture to identify any parts of the software system as good candidates for reuse in later software development efforts. You propose any modifications to the architecture or requirements that make the architecture more reusable.</p>
<b>Performers</b>	Software engineer, risk analyst
<b>Inputs</b>	<ul style="list-style-type: none"><li>• System reusability analysis results</li><li>• Software requirements</li><li>• Software system architecture</li><li>• Cycle plan</li><li>• Analysis requests</li><li>• Reuse library</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Software reusability analysis results</li><li>• Product change data</li><li>• Resource request</li></ul>
<b>Entrance Criteria</b>	<p>There is a need to analyze software reusability based on one or more of the following inputs:</p> <ul style="list-style-type: none"><li>• You have received a request to identify reusable components that the software engineers can use during the software system development.</li><li>• You have received a request to analyze the software requirements and the software architecture to identify those parts of the software system that future projects can reuse.</li><li>• You have identified a specific reusable component that may be useful to your project.</li></ul>

***Exit Criteria***

- You have received the baselined cycle development plan.
- You have documented and completed the results of the software reusability analysis.
- You have identified and qualified reusable components that the software engineers will use during software system development.
- You have proposed any changes to the software system architecture and/or software requirements that will improve the reusability of the software to the CCB.
- You have submitted requests for COTS software to project management.

***Measurables***

- The percentage of software requirements that the reusable components satisfy
- The number of changes that you proposed to the software system architecture and/or the software requirements
- The number of labor hours needed to complete reusability analysis

### **B.2.5 Analyze Software Modifiability**

**Overview**

In the Analyze Software Modifiability activity, you analyze the software system architecture to identify any changes that may make it easier to modify, improve, extend, maintain, or port the system. You should try to isolate those parts of the architecture that depend on the operating system, database management system, or target environment. If you are delivering a subset of the proposed functionality to the customer, you should ensure that you have provided proper “hooks” in the architecture so that you can expand it to accommodate the full system functionality. You should enforce the use of modular structured programming techniques.

You document the results of the analysis and propose any changes to the software requirements and/or software system architecture that will improve the modifiability of the system.

**Performers**

Software engineer, risk analyst

**Inputs**

- Software system architecture
- System modifiability analysis results
- Cycle plan
- Target environment
- Database management system
- Database description
- Analysis requests

**Outputs**

- Software modifiability analysis results
- Product change data

**Entrance Criteria**

- You have received a request to analyze the software system architecture to ensure that you can easily modify, improve, extend, maintain, and port it.
- The database description is detailed enough to analyze the software system modifiability.
- You have identified the target environment.
- You have identified the database management system.
- You have received the baselined cycle development plan.

**Exit Criteria**

- You have documented and completed the results of the software modifiability analysis.

***Measurables***

- You have proposed any changes to the software system architecture that will improve the modifiability of the software system to the CCB.
- The number of changes that you proposed to the software system architecture
- The number of labor hours needed to complete the modifiability analysis

## **B.2.6 Analyze Software Functionality**

<b>Overview</b>	In the Analyze Software Functionality activity, you analyze any incomplete, inconsistent, or ambiguous software requirements that define the functionality of the software system. If limited resources prevent you from developing the complete system functionality, work with the customer and the user to derive a subset of the functionality requirements that satisfy the project mission.
<b>Performers</b>	Software engineer, risk analyst, customer, user
<b>Inputs</b>	<ul style="list-style-type: none"><li>• System functionality analysis results</li><li>• Cycle plan</li><li>• RMP</li><li>• Software requirements</li><li>• Software system architecture</li><li>• Analysis requests</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Software functionality analysis results</li><li>• Product change data</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• There is a need to analyze the feasibility of satisfying all or part of the software functional requirements based on one or more of the following inputs:<ul style="list-style-type: none"><li>– The risk assessment has identified some functional software requirements as a high risk.</li><li>– You have received a request to analyze the functional software requirements.</li><li>– The system functionality analysis results indicate a problem with the software functionality.</li></ul></li><li>• You have received a baselined version of the software requirements.</li><li>• You have received the baselined cycle development plan.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have documented and completed the results of the software functionality analysis.</li><li>• You have proposed any changes to the software requirements that will improve the functionality of the system to the customer and user for acceptance.</li></ul>



***Measurables***

- The number of the changes that you proposed to the software requirements
- The number of labor hours needed to complete the functionality analysis

## **B.2.7 Design Database**

### ***Overview***

In the Design Database activity, you develop the database design description, which establishes the data models, schemas, and physical structure of the database. Because the database design has a major impact on the software design, complete the database design during the software requirements phase or early in the software design phase to minimize cost and schedule impact.

This activity consists of the following tasks:

- You perform local and global information flow modeling. During this task, you define the following:
  - Data flows throughout the system
  - Information models for each CSCI and for the entire system
  - Data classification, requirements, and sources
- You develop the conceptual and external schemas. During this task, you define the following:
  - Data structures for system-wide and CSCI-oriented views of the system
  - The user views of the database to support their own application idiosyncrasies
  - A logical database schema that resolves the differences between the different views
- You design the physical database. You take the conceptual schema and map it to the specific hardware and software environment. You describe the data entities, attributes, relationships, and constraints. You should analyze the physical database design to ensure that you have satisfied all relevant, nonfunctional requirements.

### ***Performers***

Software engineer

### ***Inputs***

- Software requirements
- Cycle plan
- System requirements
- Database management system
- Product change data

### ***Outputs***

The output from this activity is the database description.

***Entrance Criteria***

- There is a need to develop or modify the database description based on one or more of the following inputs:
  - A baselined set of system requirements
  - Software requirements that help specify the database description
  - An ECP that affects the database description
- You have received the baselined cycle development plan.
- You have identified the database management system.

***Exit Criteria***

The exit criterion for this activity is that the database description meets the following criteria:

- It is internally consistent to reduce the chances of contradictory results from the information system.
- It is complete to ensure that the database can satisfy known information requirements and enforce known constraints.
- It is robust to allow adaptation to foreseeable changes in the information requirements.
- It is efficient to ensure that the database management system updates, stores, and retrieves the data to be readily accessible to users.

***Measurables***

- The number of schema changes you made during this iteration through the activity
- The number, severity, and source of ECPs that affect the database description
- The number of labor hours needed to complete the database design

### **B.2.8 Design Software Components**

**Overview**

In the Design Software Components activity, you refine the software system architecture until you have identified the lowest level components and their interfaces. For each component, you specify the data structure, algorithm, and control information to a level that allows direct translation to a programming language. For each component interface, you specify the parameter formats, exception conditions, and limit values. You define and finalize all global data structures.

You file software problem reports when you find any problems or inconsistencies in the software component design.

**Performers**

Software engineer

**Inputs**

- Software requirements
- Software system architecture
- Database description
- Cycle plan
- Software engineering environment
- Reuse library
- Software problem report

**Outputs**

- Computer software unit (CSU) design
- CSU interface design
- Software problem report

**Entrance Criteria**

- There is a need to develop or modify the detailed design of a software component based on one or more of the following inputs:
  - A baselined set of software requirements
  - A baselined software system architecture
  - A baselined database description
  - A solution to a software problem report
- You have received the baselined cycle development plan.
- The software engineering environment is available to design the software components.

***Exit Criteria***

- You have identified all reusable components.
- The detailed design for each component meets the following criteria:
  - It is traceable to the software requirements.
  - It is consistent with the software architecture.
  - It is feasible.
  - It is maintainable.
  - It complies with project standards.
- You have filed all software problem reports.

***Measurables***

- The number of CSU designs that you created, modified, or deleted during this iteration through the activity
- The number of software problem reports that you generated against each CSU design
- The number of labor hours needed to complete each CSU design

**B.2.9 Create Software Components**

<b>Overview</b>	In the Create Software Components activity, you develop and document the source code for each CSU in accordance with the detailed design. Source code that you are reusing but that requires major changes should be subject to the same standards as new units. You develop and document any database utilities that you require. You file software problem reports against any problems you found during the verification of this activity.
<b>Performers</b>	Software engineer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Cycle plan</li><li>• CSU design</li><li>• CSU interface design</li><li>• Software engineering environment</li><li>• Reuse library</li><li>• Software problem report</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• CSU</li><li>• Software problem report</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• There is a need to develop or modify the source code for a software component based on one or more of the following:<ul style="list-style-type: none"><li>– The CSU interface design and CSU design for that component</li><li>– A solution to a software problem report</li></ul></li><li>• You have received the baselined cycle development plan.</li><li>• The software engineering environment is available for implementing the software components.</li><li>• You have identified all reusable components.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• The CSU:<ul style="list-style-type: none"><li>– Satisfies all allocated requirements</li><li>– Is correct, both semantically and syntactically</li><li>– Is consistent with the CSU design</li><li>– Is maintainable</li></ul></li></ul>

- Is understandable
- Complies with project standards
- You have filed all software problem reports

***Measurables***

- The number, severity, and source of software problem reports that you filed against the CSU
- The ratio of the actual CSU size to the estimated CSU size
- The number of labor hours required to code each CSU
- The ratio of the number of CSUs coded to the total number of CSUs initially identified in the detailed design

**B.2.10 Integrate Software Components**

<b>Overview</b>	In the Integrate Software Components activity, you integrate the CSUs into CSCs and CSCIs according to the software build definitions. You should use the “build a little, test a little” approach and be concerned with the construction of an ever-increasing string of functionality through a progressive, incremental process of adding CSUs to the growing core of software.
<b>Performers</b>	Software engineer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Cycle plan</li><li>• CSU</li><li>• Database management system</li><li>• Software build definition</li><li>• CSC</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• CSC</li><li>• CSCI</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• There is a need to integrate CSUs into CSCs and CSCIs based on the software build definitions.</li><li>• You have received the baselined cycle development plan.</li><li>• You have identified the database management system.</li></ul>
<b>Exit Criteria</b>	The exit criterion for this activity is that you have built all CSCIs.
<b>Measurables</b>	There are no measurements to be taken for this activity.



## **B.3 PRODUCT VERIFICATION AND VALIDATION**

The Product Verification and Validation activity class includes all activities related to the verification and validation of the system and software requirements.

### **B.3.1 Plan Verification and Validation**

<b>Overview</b>	In the Plan Verification and Validation activity, you plan the product verification and validation activities. This includes using the project master schedule, product size estimate, and available resources to develop a verification and validation plan and make resource requests. It also includes using the documentation requirements to establish verification documentation for the Product Verification and Validation activities.
<b>Performers</b>	Verification engineer, project manager
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Project master schedule</li><li>• Available resources</li><li>• Documentation requirement</li><li>• Product size estimate</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Project master schedule</li><li>• Resource request</li><li>• Verification and validation plan</li><li>• Verification documentation</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• You have a product size estimate with sufficient detail to allow estimating the resources required for Product Verification and Validation activities.</li><li>• You have received the project master schedule.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have developed the verification and validation plan.</li><li>• You have identified the resources needed for Product Verification and Validation activities and made appropriate resource requests.</li><li>• You have incorporated Product Verification and Validation activities into the project master schedule.</li><li>• You have established the verification documentation.</li></ul>
<b>Measurables</b>	<ul style="list-style-type: none"><li>• The actual cost of the product verification and validation activities</li><li>• The actual duration of the product verification and validation activities</li></ul>

## **B.4 SYSTEM VERIFICATION AND VALIDATION**

The System Verification and Validation activity class includes those activities that are concerned with the verification and validation of the system requirements.

### **B.4.1 Validate System Requirements**

<b>Overview</b>	In the Validate System Requirements activity, you validate with the customer and user that the system requirements are complete and correct. You and the customer evaluate the system requirements to make certain they meet his needs. If they do not, you negotiate a mutually acceptable change and document this with a system requirements change.
<b>Performers</b>	Project manager, systems engineer, customer, user
<b>Inputs</b>	The inputs to this activity are the system requirements.
<b>Outputs</b>	<ul style="list-style-type: none"><li>• System requirements validation results</li><li>• Product change data</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• You have a set of system requirements in a proper format, and with sufficient detail, to be validated.</li><li>• The customer has agreed to meet and validate the current system requirements.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have validated the system requirements with the customer.</li><li>• You have generated system requirements changes as necessary.</li></ul>
<b>Measurables</b>	The measurement to be taken for this activity is the number of system requirement changes, their impact (e.g., major or minor), and their cost.

## **B.4.2 Define System Integration Verification**

### ***Overview***

In the Define System Integration Verification activity, you plan the verification of the system to ensure that it meets all specified system requirements. This includes using the requirements flow diagram and verifiability checklists to define and generate system integration verification cases that map to all system requirements and documenting both the system integration verification cases and expected system integration verification results in the verification documentation. It also includes updating the verification and validation plan and the project master schedule with any new information regarding the Verify System Integration activity.

### ***Performers***

Verification engineer, systems engineer

### ***Inputs***

- Project master schedule
- System requirements
- Verifiability checklists
- Verification documentation
- Verification and validation plan
- Requirements flow diagram

### ***Outputs***

- Project master schedule
- Verification documentation
- Verification and validation plan

### ***Entrance Criteria***

- You have received a baselined version of the system requirements.
- You have filled out the requirements flow diagram from the system requirements sufficiently to define the system integration verification cases.
- You have filled out the verifiability checklists from the system requirements sufficiently to determine the expected system integration verification results.
- You have received a baselined version of the verification and validation plan.
- You have received the project master schedule.

### ***Exit Criteria***

- You have documented both the system integration verification cases and expected system integration verification results in the verification documentation.

- You have updated the verification and validation plan as necessary.
- You have updated the project master schedule as necessary.

***Measurables***

- The actual cost to perform this activity
- The actual duration of this activity

### **B.4.3 Verify System Integration**

<b>Overview</b>	In the Verify System Integration activity, you actually verify the entire integrated system. You execute all system integration verification cases, which generate actual system integration verification results. You record both the actual system integration verification results and any deviations between actual system integration verification results and expected system integration verification results in the verification log. You also generate an software problem report for each recorded deviation.
<b>Performers</b>	Verification engineer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• System</li><li>• Verification documentation</li><li>• Verification and validation plan</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Software problem report</li><li>• Verification log</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• The integrated system is ready for verification.</li><li>• You have received all system integration verification cases and expected system integration verification results.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have executed all system integration verification cases.</li><li>• You have recorded the actual system integration verification results in the verification log.</li><li>• You have recorded any deviations between actual system integration verification results and expected system integration verification results in the verification log.</li><li>• You have generated a software problem report for each recorded deviation.</li></ul>
<b>Measurables</b>	<ul style="list-style-type: none"><li>• The actual cost of this activity</li><li>• The actual duration of this activity</li><li>• The number of software problem reports you generated and their severity (e.g., fatal, serious, or mild)</li></ul>

**B.4.4 Demonstrate System Capabilities**

<b>Overview</b>	In the Demonstrate System Capabilities activity, you demonstrate the system to the customer to show that it meets all system qualification requirements. Often, the customer actually exercises the system to determine that the system qualification requirements have been fulfilled.
<b>Performers</b>	Project manager, customer, user
<b>Inputs</b>	<ul style="list-style-type: none"><li>• System qualification requirement</li><li>• Verification and validation plan</li><li>• System</li></ul>
<b>Outputs</b>	The outputs from this activity are the system demonstration results.
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• The system is ready for demonstration.</li><li>• The verification and validation plan is complete for this activity.</li><li>• You have identified the system qualification requirements.</li></ul>
<b>Exit Criteria</b>	The exit criterion for this activity is that the customer is satisfied that the system meets all system qualification requirements.
<b>Measurables</b>	The measurement to be taken for this activity is the percentage of system qualification requirements satisfied during the activity.

## **B.5 SOFTWARE VERIFICATION AND VALIDATION**

The Software Verification and Validation activity class includes activities concerned with the verification and validation of the software requirements.

### **B.5.1 Validate Software Requirements**

***Overview***

In the Validate Software Requirements activity, you validate with the customer and user that the software requirements are complete and correct. You and the customer evaluate the software requirements to make certain they meet his needs. If they do not, you negotiate a mutually acceptable change and document this with a software requirement change.

***Performers***

Software engineer, customer, project manager, user

***Inputs***

The inputs to this activity are the software requirements.

***Outputs***

- Product change data
- Software requirements validation results

***Entrance Criteria***

- You have a set of software requirements in a proper format and with sufficient detail to be validated.
- The customer has agreed to meet and validate the current software requirements.

***Exit Criteria***

- You have validated the software requirements with the customer.
- You have generated software requirements changes as necessary.

***Measurables***

The measurement to be taken for this activity is the number of software requirements changes you generated and their impact (e.g., major or minor).

## **B.5.2 Define Software Component Verification**

### ***Overview***

In the Define Software Component Verification activity, you plan the verification of a CSU to ensure that it meets all specified software requirements. This includes using the functionality flow chart and verifiability checklists to define and generate software component verification cases, which map to all specified software requirements and documenting both the software component verification cases and expected software component verification results in the verification documentation. It also includes updating the verification and validation plan and the project master schedule with any new information regarding the Verify Software Component activity.

### ***Performers***

Verification engineer, software engineer

### ***Inputs***

- Software requirements
- Verifiability checklists
- Verification documentation
- Verification and validation plan
- Project master schedule
- Requirements flow diagram

### ***Outputs***

- Project master schedule
- Verification documentation
- Verification and validation plan

### ***Entrance Criteria***

- You have received a baselined version of the software requirements.
- You have filled out the requirements flow diagram from the software requirements sufficiently to define software component verification cases.
- You have filled out the verifiability checklists from the software requirements sufficiently to determine expected software component verification results.
- You have received a baselined version of the verification and validation plan.
- You have received the project master schedule.

### ***Exit Criteria***

- You have documented both the software component verification cases and expected software component verification results in the verification documentation.



- You have updated the verification and validation plan as necessary.
- You have updated the project master schedule as necessary.

***Measurables***

- The actual cost to perform this activity
- The actual duration of this activity

### **B.5.3 Verify Software Component**

<b>Overview</b>	In the Verify Software Component activity, you actually verify the CSU. You execute all software component verification cases, which generate actual software component verification results. You record both the actual software component verification results and any deviations between actual software component verification results and expected software component verification results in the verification log. You also generate a software problem report for each recorded deviation.
<b>Performers</b>	Verification engineer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Verification documentation</li><li>• CSU</li><li>• Verification and validation plan</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Software problem report</li><li>• Verification log</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• The CSU is ready for verification.</li><li>• You have received all software component verification cases and expected software component verification results.</li></ul>
<b>Exit Criteria</b>	<p>The exit criteria for this activity are:</p> <ul style="list-style-type: none"><li>• You have executed all software component verification cases.</li><li>• You have recorded the actual software component verification results in the verification log.</li><li>• You have recorded any deviations between actual software component verification results and expected software component verification results in the verification log.</li><li>• You have generated a software problem report for each recorded deviation.</li></ul>
<b>Measurables</b>	<ul style="list-style-type: none"><li>• The actual cost of performing this activity</li><li>• The actual duration of this activity</li><li>• The number of software problem reports you generated and their severity (e.g., fatal, serious, or mild)</li></ul>

**B.5.4 Define Software Integration Verification****Overview**

In the Define Software Integration Verification activity, you plan the verification of a CSC or CSCI to ensure that it meets the software requirements. This includes using the requirements flow diagram and verifiability checklists to define and generate software integration verification cases that map to all specified software requirements and documenting both the software integration verification cases and expected software integration verification results in the verification documentation. It also includes updating the verification and validation plan and the project master schedule with any new information regarding the Verify Software Integration activity.

**Performers**

Verification engineer, systems engineer, software engineer

**Inputs**

- Project master schedule
- Software requirements
- Verifiability checklists
- Verification documentation
- Verification and validation plan
- Requirements flow diagram

**Outputs**

- Project master schedule
- Verification documentation
- Verification and validation plan

**Entrance Criteria**

- You have received a baselined version of the software requirements.
- You have filled out the requirements flow diagram from the software system architecture and software requirements sufficiently to define the software integration verification cases.
- You have filled out the verifiability checklists from the software system architecture and software requirements sufficiently to define the software integration verification cases.
- You have received a baseline version of the verification and validation plan.
- You have received the project master schedule.

**Exit Criteria**

- You have documented both the software integration verification cases and expected software integration verification results in the verification documentation.

- You have updated the verification and validation plan.
- You have updated the project master schedule.

***Measurables***

- The cost of this activity
- The duration of this activity

**B.5.5 Verify Software Integration**

<b>Overview</b>	In the Verify Software Integration activity, you actually verify the CSC or CSCI. You execute all software integration verification cases that generate actual software integration verification results. You record both the actual software integration verification results and any deviations between actual software integration verification results and expected software integration verification results in the verification log. You also generate a software problem report for each recorded deviation.
<b>Performers</b>	Verification engineer
<b>Inputs</b>	<ul style="list-style-type: none"><li>• Verification documentation</li><li>• CSC</li><li>• CSCI</li></ul>
<b>Outputs</b>	<ul style="list-style-type: none"><li>• Software problem report</li><li>• Verification log</li></ul>
<b>Entrance Criteria</b>	<ul style="list-style-type: none"><li>• The CSC or CSCI is ready for verification.</li><li>• You have received all software integration verification cases and expected software integration verification results.</li></ul>
<b>Exit Criteria</b>	<ul style="list-style-type: none"><li>• You have executed all software integration verification cases.</li><li>• You have recorded the actual software integration verification results in the verification log.</li><li>• You have recorded any deviations between actual software integration verification results and expected software integration verification results in the verification log.</li><li>• You have generated a software problem report for each recorded deviation.</li></ul>
<b>Measurables</b>	<ul style="list-style-type: none"><li>• The actual cost to perform this activity</li><li>• The actual duration of this activity</li><li>• The number of software problem reports you generated and their severity (e.g., fatal, serious, or mild)</li></ul>

## **B.6 OPERATION AND MAINTENANCE**

The Operation and Maintenance activity class includes all those activities related to system installation and operational support.

### **B.6.1 Plan System Installation**

#### ***Overview***

In the Plan System Installation activity, you analyze the system architecture, the system build definitions, and the client's need for continuity and level of service to derive a plan on how best to install the system in the target environment. The installation plan should address the following:

- The schedule for all activities
- The configuration items to be delivered to the site
- The number and qualifications of installation personnel
- The equipment installation procedures
- The hardware, software, tools, documentation, and office space required for installation
- The target facility requirements
- The anticipated effects on client and operations personnel
- The special requirements governing the movement of equipment at the site

If the developed system is to replace an existing system, develop a transition plan to minimize the disruption in ongoing operations. The transition plan should identify:

- Needed support services, such as reference documents, technical assistance, and follow-on training
- Staffing requirements for the operation and maintenance organization
- The system release process
- Data migration from the old system to the new system
- Problem identification and resolution procedures
- Final cutover procedures

#### ***Performers***

Systems engineer, technical support engineer

#### ***Inputs***

- Target environment

	<ul style="list-style-type: none"><li>• Cycle plan</li><li>• System requirements</li><li>• System architecture</li></ul>
<b><i>Outputs</i></b>	<ul style="list-style-type: none"><li>• Installation plan</li><li>• Transition plan</li></ul>
<b><i>Entrance Criteria</i></b>	<ul style="list-style-type: none"><li>• You have received the baselined system architecture.</li><li>• You have received the baselined system requirements.</li><li>• You have identified the target environment.</li><li>• You have received the baselined cycle development plan.</li></ul>
<b><i>Exit Criteria</i></b>	<ul style="list-style-type: none"><li>• The project manager and the customer approve the system installation plan.</li><li>• The project manager and the customer approve the transition plan.</li></ul>
<b><i>Measurables</i></b>	The measurement to be taken for this activity is the number of labor hours required to write and review the installation plan.

**B.6.2 Install System****Overview**

In the Install System activity, you ensure that you have prepared the operational sites for system installation. This includes ensuring that all cabling, cooling, test, and security equipment is available at the facilities.

You distribute the software, hardware, and documentation to the operational sites and install the system in the target environment according to the procedures in the system installation plan. You ensure the integrity and quality of the installed system through a series of installation tests and audits. You demonstrate that the system performs as expected in the target environment so that the customer can officially accept the system.

**Performers**

Technical support engineer

**Inputs**

- Target environment
- Cycle plan
- Operating documentation
- System
- Installation plan

**Outputs**

The output from this activity is customer acceptance.

**Entrance Criteria**

- You have received the baselined system installation plan.
- You have received the baselined cycle development plan.
- The system increment to be installed is ready to be delivered.
- All associated documentation is complete and ready to be delivered.

**Exit Criteria**

The exit criterion for this activity is that the customer has accepted the system.

**Measurables**

The measurement to be taken for this activity is the number of labor hours required to complete system installation.



### **B.6.3 Provide Operational Support**

**Overview** In the Provide Operational Support activity, you provide technical assistance in response to customer requests. You issue a software problem report for any anomalies that you detect during system operation so that you can resolve them.

You perform regular maintenance to the system to ensure that it performs efficiently. This includes tuning the system to ensure that it performs optimally, upgrading hardware or software components when needed, and installing enhancements to system capabilities.

If the system is replacing an existing system, you may provide technical consulting to smooth out the transition process.

**Performers** Technical support engineer

**Inputs**

- Transition plan
- Technical support requests
- System
- Maintenance requests
- Support environment
- Operating documentation

**Outputs** The output from this activity is the product change data.

**Entrance Criteria**

- You have distributed the operating documentation to the operational sites.
- You have installed the system at the operational sites.
- There is a need to provide technical assistance and consulting based on one of the following:
  - You have received a technical support request.
  - You have received a maintenance request.
  - The transition plan specifies that technical consulting will be provided to the customer.

**Exit Criteria**

- You have satisfied the technical support request.
- You have successfully performed system maintenance.
- You have reported any system anomalies.

***Measurables***

- The number of software problem reports you generated
- The number of labor hours required to service the system

#### **B.6.4 Identify New Operational Capabilities**

<b>Overview</b>	In the Identify New Operational Capabilities activity, to solve an operational problem or satisfy any customer requests or user requests, you analyze the current operational system and decide to add a new operational capability, enhance an existing operational capability, or define the operational capabilities for a replacement system. You base your choice of action on the cost, benefits, and risk to your customer.
<b>Performers</b>	Technical support engineer, systems engineer, customer, user
<b>Inputs</b>	<ul style="list-style-type: none"><li>• System</li><li>• Product change data</li><li>• Customer/user feedback</li></ul>
<b>Outputs</b>	The outputs from this activity are the operational capabilities.
<b>Entrance Criteria</b>	<p>There is a need to evaluate the current operational system and decide whether you should enhance it based on one of the following:</p> <ul style="list-style-type: none"><li>• There is customer or user request for new operational capabilities.</li><li>• An operational problem report has identified an operational capability that needs to be enhanced.</li></ul>
<b>Exit Criteria</b>	The exit criterion for this activity is that you have defined any new or enhanced operational capabilities.
<b>Measurables</b>	There are no measurements to be taken for this activity.

## APPENDIX C. ARTIFACT DESCRIPTIONS

This appendix provides descriptions of artifacts that the activities specified in Appendix B use or generate as inputs or outputs. Each artifact description includes a description and a list of activities for which it is used as an input or output. An activity may not directly reference each artifact as an input or output; instead, it may reference a higher level artifact, with the reference to its subartifacts assumed. In cases where artifacts are composed of other artifacts, references are provided to the lower level artifacts. Thus, the artifacts form a tree; you may traverse the tree by starting at the root. The artifacts are provided in alphabetical order.

Activities and artifacts in boldface type are not specifically listed in Appendix B but are activities defined and supported by the conceptual ESP model.

### ANALYSIS REQUESTS

<b><i>Description</i></b>	Analysis requests include requests for specific aspects or features that need to be analyzed.
<b><i>Generated by</i></b>	<ul style="list-style-type: none"><li>• Define Software and Interface Requirements</li><li>• Define System Requirements</li><li>• Design Software Architecture</li><li>• Develop System Architecture</li></ul>
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"><li>• Analyze Software Functionality</li><li>• Analyze Software Modifiability</li><li>• Analyze Software Performance</li><li>• Analyze Software Reusability</li><li>• Analyze System Dependability</li><li>• Analyze System Functionality</li><li>• Analyze System Modifiability</li><li>• Analyze System Performance</li></ul>

- Analyze System Reusability
- Design User Interface

## AVAILABLE RESOURCES

**Description** Available resources contain the resources within the organization. The project must request the use of a resource. Resources include staffing, equipment, facilities, or computer equipment. This artifact is a subartifact of **Spiral Plan Documents**.

**Generated by** The **Develop/Update Estimate of the Situation** activity generates this artifact.

**Used as Input to**

- Plan Verification and Validation
- Recommend Software System Development Strategy
- Commit to Plan
- Plan and Schedule
- Perform Risk Analysis
- Develop/Update Estimate of the Situation
- Define Approach
- Review Context
- Review Risk Analysis
- Plan Risk Aversion
- Commit to Risk Aversion Strategy
- Execute Risk Aversion
- Review Alternative
- Review Technical Product
- Review Progress
- Update Spiral Plan
- Commit to Proceed

## COMPANY IDEAS

**Description** Company ideas contain those ideas for a new or modified system that sources internal to the company or organization but external to the project generate. This artifact is a subartifact of **Supporting Document**.

<b><i>Generated by</i></b>	The <b>Define Approach</b> activity generates this artifact.
<b><i>Used as Input to</i></b>	The Specify Operational Concept activity uses this artifact as input.

## CSC

<b><i>Description</i></b>	A CSC is a distinct part of a CSCI. You may further decompose CSCs into other CSCs and CSUs.
<b><i>Generated by</i></b>	The Integrate Software Components activity generates this artifact.
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"> <li>• Integrate Software Components</li> <li>• Verify Software Integration</li> </ul>

## CSCI

<b><i>Description</i></b>	CSCI, is a configuration item for computer software.
<b><i>Generated by</i></b>	The Integrate Software Components activity generates this artifact.
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"> <li>• Integrate Software System Components</li> <li>• Product Change Control</li> <li>• Verify Software Integration</li> </ul>

## CSU

<b><i>Description</i></b>	A CSU is an element specified in the design of a CSC that is separately testable.
<b><i>Generated by</i></b>	The Create Software Components activity generates this artifact.
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"> <li>• Integrate Software Components</li> <li>• Verify Software Component</li> </ul>

## CSU DESIGN

<b><i>Description</i></b>	A CSU design is a description of the component, internal algorithms, external relationships, and data structures required for developing the CSU.
<b><i>Generated by</i></b>	The Design Software Components activity generates this artifact.
<b><i>Used as Input to</i></b>	The Create Software Components activity uses this artifact as input.

## CSU INTERFACE DESIGN

<b><i>Description</i></b>	A CSU interface design is a description of the interface and input/output formats for the CSU.
---------------------------	--

<b><i>Generated by</i></b>	The Design Software Components activity generates this artifact.
<b><i>Used as Input to</i></b>	The Create Software Components activity uses this artifact as input.

## **CUSTOMER ACCEPTANCE**

<b><i>Description</i></b>	Customer acceptance is a formal acceptance of the system by the customer after you install the system at the operational site and the customer agrees that the system performs as expected.
<b><i>Generated by</i></b>	The Install System activity generates this artifact.
<b><i>Used as Input to</i></b>	The Commit to Proceed activity uses this artifact as input.

## **CUSTOMER/USER FEEDBACK**

<b><i>Description</i></b>	Customer/user feedback represents requests for new or modified capabilities of a system. At this point, you have not formalized the request into a contract or statement of work.
<b><i>Generated by</i></b>	The Commit to Proceed activity generates this artifact.
<b><i>Used as Input to</i></b>	The Identify New Operational Capabilities activity uses this artifact as input.

## **CUSTOMER/USER REQUESTS**

<b><i>Description</i></b>	Formalized feedback documented as a modification to or an initial request documented in the contract or statement of work.
<b><i>Generated by</i></b>	An external source generates this artifact.
<b><i>Used as Input to</i></b>	The Specify Operational Concept activity uses this artifact as input.

## **CYCLE PLAN**

<b><i>Description</i></b>	A cycle plan is a controlling document that defines and schedules the activities for a project cycle. It determines the tasks to be performed for each activity, the dependency relationship between activities and tasks, and the resources allocated to the activity.
<b><i>Generated by</i></b>	<ul style="list-style-type: none"><li>• Plan and Schedule</li><li>• Commit to Plan</li></ul>
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"><li>• Plan and Schedule</li><li>• Commit to Plan</li><li>• Analyze Software Functionality</li></ul>

- Analyze Software Modifiability
- Analyze Software Performance
- Analyze Software Reusability
- Analyze System Dependability
- Analyze System Functionality
- Analyze System Modifiability
- Analyze System Performance
- Analyze System Reusability
- Create Software Components
- Define Software and Interface Requirements
- Define System Requirements
- Design Database
- Design Software Architecture
- Design Software Components
- Design User Interface
- Develop System Architecture
- Install System
- Integrate Software Components
- Integrate Software System Components
- Plan System Installation
- Recommend Software System Development Strategy

## **DATABASE DESCRIPTION**

<i>Description</i>	A database description is a description of the entities, relationships, and attributes that make up the system database.
<i>Generated by</i>	The Design Database activity generates this artifact.
<i>Used as Input to</i>	<ul style="list-style-type: none"><li>• Analyze Software Modifiability</li></ul>



- Analyze Software Performance
- Design Software Architecture
- Design Software Components

## **DATABASE MANAGEMENT SYSTEM**

**Description** A database management system controls data structures containing interrelated data stored to optimize accessibility, control redundancy, and offer multiple views of the data to multiple application programs.

**Generated by** An external source generates this artifact.

**Used as Input to**

- Analyze Software Modifiability
- Design Database
- Integrate Software Components

## **DEPENDABILITY REQUIREMENT**

**Description** Dependability requirements determine the system requirements for such reliability characteristics as mean time between failure.

**Generated by** The Define System Requirements activity generates this artifact.

**Used as Input to**

- Analyze System Dependability
- Define Software and Interface Requirements
- Define System Integration Verification
- Design Database
- Develop System Architecture
- Plan System Installation
- Recommend Software System Development Strategy
- Validate System Requirements

## **HWCI**

**Description** An HWCI is a configuration item for computer hardware.

**Generated by** An external source generates this activity.

**Used as Input to** The Integrate Software System Components activity uses this artifact as input.

## HUMAN FACTORS REQUIREMENT

<b>Description</b>	A human factors requirement specifies the CSCI requirements for interacting with humans.
<b>Generated by</b>	The Define Software and Interface Requirements activity generates this artifact.
<b>Used As Input To</b>	<ul style="list-style-type: none"> <li>Analyze Software Functionality</li> <li>Analyze Software Performance</li> <li>Analyze Software Reusability</li> <li>Define Software Component Verification</li> <li>Define Software Integration Verification</li> <li>Design Database</li> <li>Design Software Architecture</li> <li>Design Software Components</li> <li>Design User Interface</li> <li>Validate Software Requirements</li> </ul>

## INSTALLATION PLAN

<b>Description</b>	An installation plan details the process by which you integrate the system into its target environment and test it to ensure that it performs as expected.
<b>Generated by</b>	The Plan System Installation activity generates this artifact.
<b>Used as Input to</b>	The Install System activity uses this artifact as input.

## MAINTENANCE REQUESTS

<b>Description</b>	Maintenance requests are requests from the customer or user to correct system anomalies, to accommodate changes to the target environment, or to tune the system performance. This artifact is a subartifact of <b>Updated Process Drivers</b> .
<b>Generated by</b>	The Review Progress activity generates this artifact.
<b>Used as Input to</b>	The Provide Operational Support activity uses this artifact as input.

## MARKET DATA

<b>Description</b>	Market data is data from a survey or report on ideas for a new or modified system that comes from competitive analysis, new technology, and/or a change
--------------------	---

in standards or government regulations. This data might include an analysis of the current market situation, specifically identifying opportunities; threats; market capabilities; underlying assumptions; actual market conditions, including dissatisfactions and needs and the market composition; and legal, economic, and societal factors. This artifact is a subartifact of **Supporting Documents**.

**Generated by** The **Define Approach** activity generates this artifact.

**Used as Input to** The **Specify Operational Concept** activity uses this artifact as input.

## **OPERATIONAL CAPABILITIES**

**Description** Operational capabilities define the abilities or characteristics of the system in the target environment.

**Generated by** The **Identify New Operational Capabilities** activity generates this artifact.

**Used as Input to** The **Specify Operational Concept** activity uses this artifact as input.

## **OPERATIONAL CONCEPT**

**Description** The operational concept is a high-level description of the system concept. It includes a description of the system mission, roles, boundaries, external interfaces, and assumptions; a description of the target system environment; a description of the system capabilities; a description of the system operations, including modes, contingencies, scenarios, and schedules; and the system development environment.

**Generated by** The **Specify Operational Concept** activity generates this artifact.

**Used as Input to**

- Define System Requirements
- Formulate Potential Approaches

## **PERFORMANCE REQUIREMENT**

**Description** A performance requirement identifies the timing and sizing requirements of the system based on the operational concept and the hardware environment of the development and target systems.

**Generated by** The **Define System Requirements** activity generates this artifact.

**Used as Input to**

- Analyze System Performance
- Define Software and Interface Requirements
- Define System Integration Verification
- Design Database

- Develop System Architecture
- Plan System Installation
- Recommend Software System Development Strategy
- Validate System Requirements

## **PRODUCT CHANGE DATA**

**Description** Product change data includes ECPs, engineering change notices, engineering release orders, and any other information that you may associate with a change to the requirements or design. This artifact is a subartifact of **Product and Cycle Measurements and Status**.

- Generated by**
- Analyze Software Functionality
  - Analyze Software Modifiability
  - Analyze Software Performance
  - Analyze Software Reusability
  - Analyze System Dependability
  - Analyze System Functionality
  - Analyze System Modifiability
  - Analyze System Performance
  - Analyze System Reusability
  - Define Software and Interface Requirements
  - Design Software Architecture
  - Design User Interface
  - Develop System Architecture
  - Develop and Verify Product
  - Provide Operational Support
- Used as Input to**
- Define Software and Interface Requirements
  - Define System Requirements
  - Design Database

- Design Software Architecture
- Design User Interface
- Develop System Architecture
- Identify New Operational Capabilities
- Monitor and Review

## PROJECT MASTER SCHEDULE

**Description** A project master schedule is the highest level project schedule showing the project-level milestones and the major activities you are to conduct. This artifact is a subartifact of **Cycle Plan**.

**Generated by**

- Define Software Component Verification
- Define Software Integration Verification
- Define System Integration Verification
- Plan and Schedule
- Plan Verification and Validation

**Used as Input to**

- Plan and Schedule
- Define Software Component Verification
- Define Software Integration Verification
- Define System Integration Verification
- Commit to Plan
- Plan Verification and Validation
- Recommend Software System Development Strategy

## RECOMMENDED APPROACH

**Description** A recommended approach identifies the approach from the list of potential approaches that the project should adopt. You should include a justification for the reason you recommended this approach.

**Generated by**

- Develop System Architecture
- Formulate Potential Approaches

- Used as Input to***
- Define System Requirements
  - Develop System Architecture
  - Recommend Software System Development Strategy

## **REQUIREMENTS FLOW DIAGRAM**

***Description*** The requirements flow diagram indicates the input points for verifying the data structure or functional behavior of the artifacts in the system and for ensuring that they satisfy their supporting requirement(s).

- Generated by***
- Define System Requirements
  - Develop System Architecture

- Used as Input to***
- Define Software Component Verification
  - Define Software Integration Verification
  - Define System Integration Verification
  - Develop System Architecture

## **RESOURCE REQUEST**

***Description*** A resource request contains a request by the project for resources. You need resource requests for personnel, hardware, facility, or computer equipment resources. This artifact is a subartifact of **Cycle Plan**.

- Generated by***
- Plan and Schedule
  - Analyze Software Reusability
  - Plan Verification and Validation

***Used as Input to*** The **Commit to Plan** activity uses this artifact as input.

## **REUSABILITY REQUIREMENT**

***Description*** The reusability requirement includes system requirements identifying a need to reuse existing life-cycle components in the development of the system or a need to generate reusable components out of the final developed system.

***Generated by*** The Define System Requirements activity generates this artifact.

- Used as Input to***
- Analyze System Reusability
  - Define Software and Interface Requirements

- Define System Integration Verification
- Design Database
- Develop System Architecture
- Plan System Installation
- Recommend Software System Development Strategy
- Validate System Requirements

## **REUSE LIBRARY**

**Description** A reuse library is a controlled collection of reusable components that may be of value to multiple projects. Reusable components include software, design documentation, and requirements documentation.

**Generated by** An external source generates this artifact.

**Used as Input to**

- Analyze Software Reusability
- Analyze System Reusability
- Create Software Components
- Design Software Architecture
- Design Software Components

## **RMP**

**Description** An RMP formally documents the analysis regarding the risks on the project. You update this artifact each time you perform a risk analysis. Also included in this assessment is information on opportunities.

**Generated by**

- Perform Risk Analysis
- Review Risk Analysis
- Plan Risk Aversion
- Commit to Risk Aversion Strategy
- Execute Risk Aversion
- Review Alternative

**Used as Input to**

- Plan and Schedule

- **Perform Risk Analysis**
- **Review Risk Analysis**
- **Plan Risk Aversion**
- **Commit to Risk Aversion Strategy**
- **Execute Risk Aversion**
- **Review Alternative**
- **Analyze Software Functionality**
- **Analyze Software Performance**
- **Analyze System Dependability**
- **Analyze System Performance**
- **Define Software and Interface Requirements**
- **Define System Requirements**
- **Design Software Architecture**
- **Develop System Architecture**
- **Formulate Potential Approaches**

## **SOFTWARE ANALYSIS DATA**

**Description** Software analysis data contains the results of all the information gathering or risk mitigation activities that you performed during software development.

- Subartifacts**
- Software functionality analysis results
  - Software modifiability analysis results
  - Software performance analysis results
  - Software reusability analysis results

## **SOFTWARE BUILD DEFINITION**

**Description** A software build definition is a description of the CSUs and CSCs, as well as their interfaces, needed to develop the CSC and CSCI builds to be used for software system component integration.



- Generated by** The Design Software Architecture activity generates this artifact.
- Used as Input to** The Integrate Software Components activity uses this artifact as input.

## SOFTWARE ENGINEERING ENVIRONMENT

**Description** A software engineering environment is a set of automated tools, firmware devices, and hardware necessary to perform the software engineering effort. The automated tools may include but are not limited to compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, test tools, documentation tools, and database management system(s). The software engineering environment also supports systems engineering activities, firmware programming activities, and software programming activities.

**Generated by** An external activity generates this artifact.

- Used as Input to**
- Analyze System Functionality
  - Create Software Components
  - Define Software and Interface Requirements
  - Design Software Architecture
  - Design Software Components
  - Develop System Architecture

## SOFTWARE FUNCTIONALITY ANALYSIS RESULTS

**Description** Software functionality analysis results contain the results of the analysis that examined the feasibility of developing the functions specified by the software functional requirements.

**Generated by** The Analyze Software Functionality activity generates this artifact.

- Used as Input to**
- Define Software and Interface Requirements
  - Design Software Architecture

## SOFTWARE MODIFIABILITY ANALYSIS RESULTS

**Description** Software modifiability analysis results contain the recommendations to enhance or change the software architecture so that you improve the ease with which you can modify the software to correct faults, improve performance, or add new features or capabilities.

**Generated by** The Analyze Software Modifiability activity generates this artifact.

**Used as Input to**

- Define Software and Interface Requirements
- Design Software Architecture

## **SOFTWARE PERFORMANCE ANALYSIS RESULTS**

**Description** Software performance analysis results contain the results of the analysis that examined the speed, accuracy, and memory usage requirements for the software.

**Generated by** The Analyze Software Performance activity generates this artifact.

**Used as Input to**

- Define Software and Interface Requirements
- Design Software Architecture

## **SOFTWARE PROBLEM REPORT**

**Description** A software problem report is a document that you use to report software problems, document the analysis of the problem, and subsequently propose a solution to the problem. This artifact is a subartifact of **Product and Cycle Measurements and Status**.

**Generated by**

- Create Software Components
- Design Software Components
- Verify Software Component
- Verify Software Integration
- Verify System Integration

**Used as Input to**

- Create Software Components
- Design Software Components
- Monitor and Review

## **SOFTWARE REQUIREMENTS**

**Description** Software requirements contain those entities that define software requirements entities.

**Generated by** The Define Software and Interface Requirements activity generates this artifact.

- Used as Input to***
- Analyze Software Functionality
  - Analyze Software Performance
  - Analyze Software Reusability
  - Define Software Component Verification
  - Define Software Integration Verification
  - Design Database
  - Design Software Architecture
  - Design Software Components
  - Validate Software Requirements

## **SOFTWARE REQUIREMENTS VALIDATION RESULTS**

- Description*** Software requirements validation results document the validity of the software requirements.
- Generated by*** The Validate Software Requirements activity generates this artifact.
- Used as Input to*** An external source uses this artifact as input.

## **SOFTWARE REUSABILITY ANALYSIS RESULTS**

- Description*** Software reusability analysis results contain the results of the analysis that examine the reusable components that you should use to develop the software system and recommendations on how you need to change the software architecture to accommodate the component. Analysis results also include any requirements to which the software system must conform so that certain new components will be reusable in future software systems.
- Generated by*** The Analyze Software Reusability activity generates this artifact.
- Used as Input to***
- Define Software and Interface Requirements
  - Design Software Architecture

## **SOFTWARE SYSTEM ARCHITECTURE**

- Description*** A software system architecture describes all of the software components that satisfy the system requirements.

**Generated by** The Design Software Architecture activity generates this artifact.

**Used as Input to**

- Analyze Software Functionality
- Analyze Software Modifiability
- Analyze Software Performance
- Analyze Software Reusability
- Design Software Components
- Develop Configuration Identification

## **SOFTWARE SYSTEM DEVELOPMENT STRATEGY**

**Description** A software system development strategy describes the strategy by which you build the system, for example, the selected process model, make versus buy decisions, and the strategy by which you will deliver the system to the customer, such as through incremental system builds. This artifact is a subartifact of **Approved Risk Management Plan**.

**Generated by** The Recommend Software System Development Strategy activity generates this artifact.

**Used as Input to**

- Define System Requirements
- Develop System Architecture
- Plan and Schedule

## **SUPPORT ENVIRONMENT**

**Description** You use a support environment to maintain the system.

**Generated by** An external source generates this artifact.

**Used as Input to** The Provide Operational Support activity uses this artifact as input.

## **SYSTEM**

**Description** A system is a group of units that form a whole and operate in unison.

**Generated by** The Integrate Software System Components activity generates this artifact.

**Used as Input to** The following activities use this artifact as input:

- Demonstrate System Capabilities
- Identify New Operational Capabilities

- Install System
- Provide Operational Support
- Verify System Integration

## SYSTEM ANALYSIS RESULTS

**Description** System analysis results contain the results of all the information gathering or risk mitigation activities that you perform during system development.

- Subartifacts**
- Requirements flow diagram
  - System dependability analysis results
  - System functionality analysis results
  - System modifiability analysis results
  - System performance analysis results
  - System reusability analysis results
  - System user interface design

## SYSTEM ARCHITECTURE

**Description** A system architecture includes all system and software design components.

- Subartifacts**
- Hardware/firmware architecture
  - Hardware/software interfaces
  - System user interface design

**Generated by** The Develop System Architecture activity generates this artifact.

- Used as Input to**
- Analyze System Dependability
  - Analyze System Functionality
  - Analyze System Modifiability
  - Analyze System Performance
  - Analyze System Reusability
  - Design Software Architecture
  - Design User Interface

- Integrate Software System Components
- Plan System Installation
- Recommend Software System Development Strategy

## **SYSTEM BUILD DEFINITION**

<b>Description</b>	A system build definition is a description of the CSCIs and HWCIs, as well as their interfaces, needed to develop a system build to use for system verification and final delivery to the customer.
<b>Generated by</b>	The Recommend Software System Development Strategy activity generates this artifact.
<b>Used as Input to</b>	The Integrate Software System Components activity uses this artifact as input.

## **SYSTEM CAPABILITY**

<b>Description</b>	A system capability is an ability or functionality of the system. A system capability describes what the system is supposed to do, not how it is supposed to do it.
<b>Generated by</b>	The Specify Operational Concept activity generates this artifact.
<b>Used as Input to</b>	<ul style="list-style-type: none"><li>• Formulate Potential Approaches</li><li>• Define System Requirements</li></ul>

## **SYSTEM DEMONSTRATION RESULTS**

<b>Description</b>	System demonstration results document the results of the system validation.
<b>Generated by</b>	The Demonstrate System Capabilities activity generates this artifact.
<b>Used as Input to</b>	An external source activity uses this artifact.

## **SYSTEM DEPENDABILITY ANALYSIS RESULTS**

<b>Description</b>	System dependability analysis results contain information on the reliability, maintainability, and availability of the system.
<b>Generated by</b>	The Analyze System Dependability activity generates this artifact.
<b>Used as Input to</b>	<ul style="list-style-type: none"><li>• Define System Requirements</li><li>• Develop System Architecture</li></ul>

## **SYSTEM FUNCTIONALITY ANALYSIS RESULTS**

<b>Description</b>	System functionality analysis results contain the analysis data from analyzing the system architecture for such functionality characteristics as range of applicability, scope of automation, and conformance to standards.
--------------------	---

**Generated by** The Analyze System Functionality activity generates this artifact.

**Used as Input to**

- Define System Requirements
- Develop System Architecture

## **SYSTEM MODIFIABILITY ANALYSIS RESULTS**

**Description** System modifiability analysis results contain information on the measures of the system's ability to be improved, extended, or ported during the system's lifetime.

**Generated by** The Analyze System Modifiability activity generates this artifact.

**Used as Input to**

- Analyze Software Modifiability
- Define System Requirements
- Develop System Architecture

## **SYSTEM PERFORMANCE ANALYSIS RESULTS**

**Description** System performance analysis results provide information on the performance of individual system components and the performance of the system as a whole.

**Generated by** The Analyze System Performance activity generates this artifact.

**Used as Input to**

- Analyze Software Performance
- Define System Requirements
- Develop System Architecture

## **SYSTEM QUALIFICATION REQUIREMENT**

**Description** A system qualification requirement specifies how you will validate the system and verify the system development process.

**Generated by** The Define System Requirements activity generates this artifact.

**Used as Input to** The Demonstrate System Capabilities activity uses this artifact as input.

## **SYSTEM REQUIREMENTS**

**Description** System requirements contain artifacts that define system requirements.

**Generated by** The Define System Requirements activity generates this artifact.

- Used as Input to***
- Analyze System Functionality
  - Analyze System Modifiability
  - Analyze System Reusability
  - Define Software and Interface Requirements
  - Define System Integration Verification
  - Design Database
  - Develop System Architecture
  - Plan System Installation
  - Recommend Software System Development Strategy
  - Validate System Requirements

## **SYSTEM REQUIREMENTS VALIDATION RESULTS**

- Description*** System requirements validation results document the validity of the system requirements.
- Generated by*** The Validate System Requirements activity generates this artifact.
- Used as Input to*** An external source uses this artifact as input.

## **SYSTEM REUSABILITY ANALYSIS RESULTS**

- Description*** System reusability data contains information on what reusable components you should incorporate into the system during implementation and any recommendations for how the system should accommodate the component. Also included are how you could use and possibly modify all or part of the system architecture or the system requirements as reusable components on other systems.
- Generated by*** The Analyze System Reusability activity generates this artifact.
- Used as Input to***
- Analyze Software Reusability
  - Define System Requirements
  - Develop System Architecture

## **SYSTEM USER INTERFACE DESIGN**

- Description*** A system user interface design contains those design approaches developed by the systems engineers to demonstrate possible approaches to developing the user interface.



<b><i>Generated by</i></b>	<ul style="list-style-type: none"><li>• Design User Interface</li><li>• Develop System Architecture</li></ul>
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"><li>• Analyze System Dependability</li><li>• Analyze System Functionality</li><li>• Analyze System Modifiability</li><li>• Analyze System Performance</li><li>• Analyze System Reusability</li><li>• Design Software Architecture</li><li>• Design User Interface</li><li>• Develop System Architecture</li><li>• Integrate Software System Components</li><li>• Plan System Installation</li><li>• Recommend Software System Development Strategy</li></ul>

## TARGET ENVIRONMENT

<b><i>Description</i></b>	A target environment is the environment in which a system must operate. This artifact is a subartifact of <b>Supporting Documents</b> .
<b><i>Generated by</i></b>	An external source generates this artifact.
<b><i>Used as Input to</i></b>	<ul style="list-style-type: none"><li>• Analyze Software Modifiability</li><li>• Analyze System Dependability</li><li>• Analyze System Performance</li><li>• Design User Interface</li><li>• Develop System Architecture</li><li>• Formulate Potential Approaches</li><li>• Install System</li><li>• Plan System Installation</li></ul>

## TECHNICAL SUPPORT REQUESTS

<b><i>Description</i></b>	Technical support requests are requests from the customer or user to provide help in interpreting system output, provide support in performing complex system operations, or enhance current system features.
---------------------------	---

***Generated by*** An external source generates this artifact.

***Used as Input to***

- Provide Operational Support
- Specify Operational Concept

## **TRANSITION PLAN**

***Description*** A transition plan details the process by which you will replace an existing system by a new system. The plan describes the process for system installation, training, and operation so that there is minimal disruption in the customer's day-to-day activities. This artifact is a subartifact of **Spiral Plan**.

***Generated by*** The Plan System Installation activity generates this artifact.

***Used as Input to***

- Define Approach
- Develop/Update Estimate of the Situation
- Review Context
- Perform Risk Analysis
- Plan Risk Aversion
- Commit to Risk Aversion Strategy
- Review Alternative
- Plan and Schedule
- Commit to Plan
- Develop and Verify Product
- Monitor and Review
- Review Technical Product
- Review Progress
- Update Spiral Plan
- Commit to Proceed
- Provide Operational Support

## **VERIFIABILITY CHECKLISTS**

***Description*** Verifiability checklists trace verification cases to requirements and document the criteria used for verifying the requirements. You create one checklist for each requirement.

- |                                |  |
|--------------------------------|--|
| <b><i>Generated by</i></b>     | <ul style="list-style-type: none"><li>• Define System Requirements</li><li>• Develop System Architecture</li></ul>   |
| <b><i>Used as Input to</i></b> | <ul style="list-style-type: none"><li>• Define Software Component Verification</li><li>• Define Software Integration Verification</li><li>• Define System Integration Verification</li><li>• Develop System Architecture</li></ul> |

## **VERIFICATION AND VALIDATION PLAN**

- |                                |  |
|--------------------------------|--|
| <b><i>Description</i></b>      | A verification and validation plan outlines the resources, procedures, and approach that you will use to verify and validate system artifacts.   |
| <b><i>Generated by</i></b>     | <ul style="list-style-type: none"><li>• Define Software Component Verification</li><li>• Define Software Integration Verification</li><li>• Define System Integration Verification</li><li>• Plan Verification and Validation</li></ul>  |
| <b><i>Used as Input to</i></b> | <ul style="list-style-type: none"><li>• Define Software Component Verification</li><li>• Define Software Integration Verification</li><li>• Define System Integration Verification</li><li>• Demonstrate System Capabilities</li><li>• Verify Software Component</li><li>• Verify System Integration</li></ul> |

## **VERIFICATION DOCUMENTATION**

- |                            |   |
|----------------------------|---|
| <b><i>Description</i></b>  | Verification documentation contains information used during verification activities and includes verification cases, expected verification case results, and the verification log.  |
| <b><i>Generated by</i></b> | <ul style="list-style-type: none"><li>• Define Software Component Verification</li><li>• Define Software Integration Verification</li><li>• Define System Integration Verification</li><li>• Plan Verification and Validation</li></ul> |

- Used as Input to***
- Define Software Component Verification
  - Define Software Integration Verification
  - Define System Integration Verification
  - Verify Software Component
  - Verify Software Integration
  - Verify System Integration

## **VERIFICATION LOG**

***Description***      A verification log documents the output produced by verifying artifacts and includes actual verification case results and deviations.

- Generated by***
- Verify Software Component
  - Verify Software Integration
  - Verify System Integration

- Used as Input to***
- Define Software Component Verification
  - Define Software Integration Verification
  - Define System Integration Verification
  - Verify Software Component
  - Verify Software Integration
  - Verify System Integration

*This page intentionally left blank.*

## **LIST OF ABBREVIATIONS AND ACRONYMS**

<b>CCB</b>	<b>Configuration Control Board</b>
<b>CMM</b>	<b>Capability Maturity Model</b>
<b>COTS</b>	<b>commercial off-the-shelf</b>
<b>CSC</b>	<b>computer software component</b>
<b>CSCI</b>	<b>computer software configuration items</b>
<b>CSU</b>	<b>computer software unit</b>
<b>ECP</b>	<b>Engineering Change Proposal</b>
<b>EoS</b>	<b>Estimate of the Situation</b>
<b>ESP</b>	<b>Evolutionary Spiral Process</b>
<b>ETVX</b>	<b>Entry-Task-Validation-eXit</b>
<b>HWCI</b>	<b>hardware configuration item</b>
<b>IDEF</b>	<b>Integrated Computer-Aided Manufacturing Definition</b>
<b>IE</b>	<b>Improvement Efforts</b>
<b>OPD</b>	<b>Organizational Process Development</b>
<b>PAD</b>	<b>Project Application Development</b>
<b>PAL</b>	<b>Process Asset Library</b>
<b>PERT</b>	<b>Program Evaluation and Review Technique</b>
<b>PLD</b>	<b>Product-Line-Based Product and Process Development</b>
<b>RMP</b>	<b>risk management plan</b>
<b>SEI</b>	<b>Software Engineering Institute</b>
<b>SEPG</b>	<b>Software Engineering Process Group</b>
<b>SOW</b>	<b>statement of work</b>

STARS	Software Technology for Adaptable Reliable Systems
VCOE	Virginia Center of Excellence for Software Reuse and Technology Transfer
WBS	work breakdown structure

## **GLOSSARY**

<b>Abstraction</b>	A description of a collection of things that applies equally well to any one of them.
<b>Activity</b>	A step of a process for producing or evaluating data elements to satisfy objectives supporting that process. An activity comprises other steps.
<b>Alternative</b>	One possible way to satisfy an objective.
<b>Business area</b>	A coherent market characterized by potential customers possessing similar needs.
<b>Capability Maturity Model (CMM)</b>	A model developed by the Software Engineering Institute used to assess organizational software process maturity.
<b>Change control</b>	The managed evaluation, coordination, approval or disapproval, and implementation of changes to work products.
<b>Commercial off-the-shelf (COTS)</b>	A ready-made product that is generally available for sale.
<b>Commit</b>	To bind or obligate (oneself, one's project, or one's organization) to the consequences of a decision.
<b>Configuration management</b>	The process of identifying and defining the configuration items in a system, controlling the release and change of these items throughout the system life cycle, recording and reporting the status of configuration items and change requests, and verifying the completeness and correctness of configuration items.
<b>Context</b>	The scope in which an analysis or commitment is made. The context establishes which influences are internal (stakeholders) and external to the decision.
<b>Contract manager</b>	A person who manages the project costs and schedules for the company.



Constraint	A limitation on decisions.
Corporate manager	The person with overall profit-or-loss responsibility for the corporate division that is running the project.
Customer	The person or organization that specifies the requirements and accepts and authorizes payment for a product.
Cycle	A complete traversal of the activities specified in the ESP model, which denotes that the product has advanced by a specified amount toward its next milestone.
Data element	(1) A piece or collection of information that is used as an input or output of an activity. (2) A collection of information fundamental to a system.
Decision	A choice among allowable alternatives.
Domain	A product family and an associated production process supporting a product line.
Engineer	A person who performs technical activities for a project.
Entrance criteria	Conditions that must be met before an activity can be started.
Event driven	Indicating that progress is measured by successfully reaching successive milestones in the product life cycle and not just by cost and schedule considerations.
Evolutionary life cycle	A development approach in which functional versions of a product are built, delivered, and used. The definition of a later version is influenced by experience from use of the previous version.
Exit criteria	Conditions that must be met before an activity can be considered successfully completed.
Family	A set of things that have enough in common that it pays to consider their common characteristics before noting specific properties of instances.
Goal	A specific, time-related, measurable target.

---

Instantiation	Creating a thing from a representation of an abstraction denoting a set of such things.
Life cycle	A sequence of distinct states of an entity, beginning with its initial conception and ending when it is no longer available for use.
Method	Guidance and criteria that prescribe a systematic, repeatable technique for performing an activity.
Methodology	An integrated body of principles, practices, and methods that prescribe the proper performance of a process.
Metrics	The process and product measurements and experiences collected during a project.
Milestone	One in the sequence of recognized product life-cycle states. Milestones are used for planning and tracking progress. The milestone has been reached when the product is shown to be in the desired state.
Model	A representation of a thing from which analysis provides approximate answers to designated questions about the thing itself.
Objective	The intended or desired result of a course of action.
Organization	A unit within a company or other entity (e.g., government agency or branch of service) within which projects are managed.
Plan	A designation of tasks and resource allocations for accomplishing a specified objective.
Problem	A situation that, if not corrected, will lead to undesirable results. The purpose of risk management is to anticipate problems and deal with their root causes.
Process	A partially ordered set of steps intended to accomplish specified objectives.
Process assets	A subset of a process definition or model that depicts some combination of roles, resources, activities, constraints, or other process elements in a manner that supports the construction of complete, integrated representations of processes and activities.

---

Process driver	A characteristic that has a significant influence on the definition or instantiation of a process.
Process engineering	The construction of a process appropriate to accomplish the objectives of an organization or project.
Product	The aggregation of all work products resulting from a process or activity.
Product line	A collection of existing and potential products that address a designated business area.
Program	(1) An aggregation of software components that operates as a unit when integrated with hardware. (2) A directed, funded effort to acquire, develop, or maintain a product.
Project	An undertaking requiring concerted effort that is focused on developing or maintaining a specific product. Typically, a project has its own funding, cost accounting, and delivery schedule.
Project manager	A program, project, or staff member responsible for the management of a project. Also, a person directly responsible for the definition, cost, and schedule of a product.
Quality manager	A person who is responsible for validating or verifying the correctness or adequacy of a product or process.
Resource	Something or someone that can be used to perform a work assignment.
Risk	A potential for incurring undesirable results.
Risk analyst	A person who is trained in risk analysis and risk aversion.
Spiral	One or more cycles combined to achieve a milestone.
Stakeholder	An individual with a vested interest in a project or its outcome, such as a team member, client, or manager.
State (of a product)	(1) The condition of the product at a given instant as described by a set of characteristic variables. (2) The values assumed at a given instant by the variables that define the characteristics of the product.

Step	Either an activity or an unelaborated action.
Success criteria	The minimum acceptable work that must be accomplished in a cycle for the project to make progress.
System	A collection of hardware, software, and people that operate together to accomplish a mission.
Task	A work assignment (i.e., subject to management accountability) to accomplish a specified objective.
Taxonomy	A system of classification.
User	The person or organization that will use the system for its intended purpose when it is deployed in its environment.
Validation	The evaluation of a work product to determine whether it satisfies customer needs.
Verification	The evaluation of a work product to determine whether it meets its specification.
Waterfall life cycle	A development approach in which the product is built by successively refining from abstract to concrete. All components of the product are simultaneously developed to a similar level of detail.
Work product	Any configuration-managed artifact that is the embodiment of some data element.

*This page intentionally left blank.*

## REFERENCES

- Agresti, William  
1986  
"Conventional Software Life-Cycle Model: Its Evolution and Assumptions." *IEEE Tutorial: New Paradigms for Software Development*. Los Angeles, California: IEEE Computer Society Press.
- Boehm, Barry W.  
1986  
A Spiral Model of Software Development and Enhancement. *ACM Software Engineering Notes* 11:22-42.
- 1988  
A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21:61-72.
- 1989  
*Tutorial: Software Risk Management*. Washington, D.C.: IEEE Computer Society Press.
- Boehm, Barry W.,  
and Rony Ross  
1989  
"Theory W Software Project Management: Principles and Examples." In *Tutorial: Software Risk Management*. Washington, D.C.: IEEE Computer Society Press.
- Charette, Robert N.  
1989  
*Software Engineering Risk Analysis and Management*. New York, New York: Intertext Publications, McGraw-Hill.
- 1990  
*Applications Strategies for Risk Analysis*. New York, New York: Intertext Publications, McGraw-Hill.
- Curtis, Bill, Marc I. Kellner,  
and Jim Over  
1992  
Process Modeling. *Communications of the ACM* 35, 9:75-90.
- Feiler, Peter H. and  
Watts S. Humphrey  
1992  
*Software Process Development and Enactment: Concepts and Definitions*, CMU/SEI-92-TR-4, Draft. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute.
- Gilb, Tom  
1988  
*Principles of Software Engineering Management*. Wokingham, England: Addison-Wesley.
- Kasunic, M.D., J.W. Armitage,  
P.G. Arnold, L.P. Gates, M.I.  
Kellner, W. Moseley, K.Y.  
Nieng, J.W. Orer, R.W. Phillips,  
and J.R. Pixton  
1992  
*Summary Report for the Process Definition Advisory Group*, Special Report, SEI-92-SR-16. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute.

- Krasner, Herb, Jim Terrel, Adam Linehan, Paul Arnold, and William H. Ett  
1992  
Lessons Learned From a Software Process Modeling System. *Communications of the ACM* 35, 9:91-100.
- Over, James W.  
1991  
"STARS '91 Process Asset Library." In *Proceedings, STARS '91*. December 3-4, 1991.
- Paulk, M.C., Bill Curts, Mary Beth Chrissis, and Charles V. Weber  
1993  
*Key Practices of the Capability Maturity Model*, version 1.1, CMU/SEI-93-TR-25. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute.
- Radice, Ronald A., and Richard W. Phillips  
1988  
*Software Engineering: An Industrial Approach*. Vol. 1. Englewood Cliffs, New Jersey: Prentice-Hall.
- Snyder, James C., and Anthony J. Catanese  
1979  
*Introduction to Architecture*. New York, New York: McGraw-Hill.
- SofTech, Inc.  
1981  
*IDEF Users Manual-Function Modeling (IDEF0)*. Distributed by IDEF Users Group, Kettering, Ohio.
- Software Engineering Institute  
1992  
*STARS/SEI Process Asset Library (PAL)*, version 2.0. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute.
- Software Productivity Consortium  
1992a  
*Process Definition and Modeling Guidebook*, SPC-92041-CMC, version 01.00.02. Herndon, Virginia: Software Productivity Consortium.
- 1992b  
*Software Measurement Guidebook*, SPC-91060-CMC. Herndon, Virginia: Software Productivity Consortium.
- 1993a  
*Managing Process Improvement: A Guidebook for Implementing Change*, SPC-93105-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.
- 1993b  
*Reuse Adoption Guidebook*, SPC-92051-CMC, version 02.00.05. Herndon, Virginia: Software Productivity Consortium.
- 1993c  
*Reuse-Driven Software Processes Guidebook*, SPC-92019-CMC, version 02.00.03. Herndon, Virginia: Software Productivity Consortium.

1993d

*Using New Technologies: A Technology Transfer Guidebook*, SPC-92046-CMC, version 02.00.08. Herndon, Virginia: Software Productivity Consortium.

U.S. Air Force  
1988

*Acquisition Management: Software Risk Abatement*, AFSC/AFLCP 800-45. Washington, D.C.: Andrews Air Force Base. Air Force Systems Command and Air Force Logistics Command.



*This page intentionally left blank.*

## BIBLIOGRAPHY

### General

Andriole, Stephen J. *Storyboard Prototyping: A New Approach to User Requirements Analysis*. Wellesley, Massachusetts: QED Information Sciences, Inc., 1989.

———. *Rapid Applications Prototyping: Storyboarding for User Requirements Analysis*. Wellesley, Massachusetts: QED Information Sciences, Inc., 1991.

Ashley, David B. "Project Risk Identification Using Inference Subjective Expert Assessment and Historical Data." In *The State of the Art in Project Risk Management, Proceedings of the INTERNET International Expert Seminar in Connection With the PMI/INTERNET Joint Symposium, Atlanta, October 12-13, 1989*. Zurich, Switzerland: International Project Management Association, 9-25, 1990.

Balzer, R.T., and C. Green Cheatham. *Software Technology in the 1990s: Using a New Paradigm*. IEEE Computer (1983):39-45.

Basili, Victor R. *Use of Ada for FAA's Advanced Automation System (AAS)*, MTR-87W77. The MITRE Corporation, 1987.

Basili, Victor R., B. Boehm, J. Clapp, D. Gaumer, M. Holden, A. Salwen, and J. Summers. *Use of Ada for FAA's Advanced Automation System (AAS)*. McLean, Virginia: The MITRE Corporation. MTR-87W77. 1987.

Basili, V.R., and D.M. Weiss. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering* SE-10, 6, 1984.

Beizer, Boris. *Software System Testing and Quality Assurance*. New York, New York: Van Nostrand Reinhold Company, 1984.

———. *Software Testing Techniques*. New York, New York: Van Nostrand Reinhold Company, 1990.

Boehm, Barry W. *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

Boehm, Barry, and Frank Belz. Applying Process Programming to the Spiral Model. In *Proceedings of the 4th International Process Workshop*, May 1988.

———. Experiences With the Spiral Model as a Process Model Generator. In *Proceedings of the 4th International Software Process Workshop*, May 1988.

Boyd, Harper W., Jr., Ralph Westfall, and Stanley F. Stasch. *Marketing Research: Text and Cases*, 7th ed. Boston, Massachusetts: Irwin, 1989.

Bruce, P., and S. Pederson. *The Software Development Project: Planning and Management*. New York, New York: John Wiley & Sons, 1982.

Carr, Marvin J., Suresh L. Konda, Ira Monarch, F. Carol Ulrich, and Clay F. Walker. *Taxonomy-Based Risk Identification*, CMU/SEI-93-TR-06. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute, June, 1993.

Charette, Robert N. Risk Management Seminar. Herndon, Virginia: Software Productivity Consortium, 1991.

Cohen, William A. *High-Tech Management*. New York, New York: American Management Association, 1986.

Conte, S.D., H.E. Dunsmore, and V.Y. Shen. *Software Engineering Metrics and Models*. Menlo Park, California: Benjamin/Cummings, 1986.

Cruickshank, R.D. *A Course in System and Software Cost Engineering*. Manassas, Virginia: IBM Federal Systems Division, 1988.

Cruickshank, R.D., and M. Lesser. "An Approach to Estimating and Controlling Software Development Costs." *The Economics of Data Processing*. New York, New York: Wiley, 1982.

Curtis, Bill, Marc I. Kellner, and Jim Over. Process Modeling, *Communications of the ACM*, 35, 9:75-90, September 1992.

Curtis, B., H. Krasner, Vincent Shen, and Neil Iscoe. On Building Software Process Models Under the Lamppost, *Proceedings of the 9th International Conference on Software Engineering*, IEEE, Computer Society Washington, D.C., 96-103, 1987.

Defense Systems Management College. *Systems Engineering Management Guide*. Washington, D.C.: U.S. Government Printing Office, 1990.

Department of Defense. *Transition From Development to Production*, DOD 4245.7. Washington, D.C.: Department of Defense, 1982.

———. *Military Standard: Defense System Software Development*, DOD-STD-2167A. Washington, D.C.: Department of Defense, 1988.

———. *Defense Acquisition Management Policies and Procedures*, Department of Defense Instruction 5000.2. Washington, D.C.: Department of Defense, 1991.

Fagan, M.E. "Design and Code Inspections to Reduce Errors in Program Development." *Tutorial Software Quality Assurance: A Practical Approach*. Edited by T.S. Chow. Silver Spring, Maryland: IEEE Computer Society Press. 297-325, 1984.

Farrell, Paul V., Stuart F. Heinritz, and Clifton L. Smith. *Purchasing: Principles and Applications*. 7th ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1986.

Fournier, Roger. *Practical Guide to Structured System Development and Maintenance*. Englewood Cliffs, New Jersey: Yourdon Press, 1991.

Gaffney, J.E., Jr. "Approaches to Estimating and Controlling Software Costs." 1983 *International Conference of the Computer Measurement Group*, CMG XIV. Washington, D.C., 1983.

———. Estimation of Software Code Size Based on Quantitative Aspects of Function (With Application of Expert System Technology). *Journal of Parametrics* 4, 3:23, 1984.

Gaffney, J.E., Jr., and R. Werling. *Estimating Software Size From Counts of Externals, A Generalization of Function Points*, SPC-91094-N. Herndon, Virginia: Software Productivity Consortium, and ISPA'91, New Orleans, Louisiana, 1991.

Gildersleeve, Thomas Robert. *Successful Data Processing System Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

Hazel, M.D., L.L. Krumm, W.P. Needham, and R.C. Slate. *BCAG Avionics Computer Software Technical Standard* (D6-35071-1, Revision D). Seattle, Washington: Boeing Commercial Airplane Group, 1990.

Heirs, Ben J. *The Professional Decision-Thinker*. New York, New York: Dodd, Mead & Company. (Originally published, Great Britain: Sidgwick & Jackson, 1987.)

Helmer, O. *Social Technology*. New York, New York: Basic Books, 1966.

Hollocker, C.P. *Software Reviews and Audits Handbook*. New York, New York: John Wiley & Sons, 1990.

Humphrey, W.S. et al. *A Method for Assessing the Software Engineering Capability of Contractors*, CMU/SEI-87-TR-23, ADA 187230. Pittsburgh, Pennsylvania: Software Engineering Institute, 1987.

Humphrey, Watts S. *Characterizing the Software Process: A Maturity Framework*, CMU/SEI-87-TR-11, ESD-TR-87-112. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute, 1987.

———. *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley, 1989.

IEEE. *IEEE Software Engineering Standards Collection*, Spring Edition. New York, New York: IEEE, Inc., 1991.

IEEE Computer Society. *IEEE Standard for Developing Software Life Cycle Processes*, IEEE STD 1074. New York, New York: IEEE Computer Society, 1992.

IEEE Computer Society. *Standard for Developing Software Life Cycle Processes*, Working Draft P1074/D6, New York, New York: Institute of Electrical and Electronics Engineers, Inc., January 1, 1989.

IEEE Computer Society. *Standard for Developing Software Life Cycle Processes* (Preliminary P1074/D6) Technical Committee on Software Engineering of the IEEE Computer Society, 1991.

Ince, David. *Software Engineering*. London: Van Nostrand Reinhold (International) Co. Ltd., 1989.

International Standards Organization. *Information Technology Software Life-Cycle Process*, ISO/IEC (JTC1)-SC7, Working Draft (3). International Standards Organization, January 29, 1991.

- Kellner, Marc I. *Software Process Modeling: Value and Experience*. SEI Technical Review (1989):23-54.
- King, David. *Current Practices in Software Development*. New York, New York: Yourdon Press, 1984.
- Kloppenborg, Tim, and Samuel J. Mantel, Jr. Tradeoffs on Projects: They May Not Be What You Think. *Project Management Journal* 1:13-20, 1990.
- Koontz, Harold, Cyril O'Donnell, and Heinz Weihrich. *Management*. 8th ed. New York, New York: McGraw-Hill, 1984.
- Lazzaro, V., ed. *Systems and Procedures: A Handbook for Business and Industry*. 2nd ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1968.
- Merkhofer, Miley W. Quantifying Judgmental Uncertainty: Methodology, Experiences, and Insights. *IEEE Transactions on Man, Systems, and Cybernetics* SMC-17, 5, 1987.
- Miser, Hugh J., and Edward S. Quade. *Handbook of Systems Analysis: Overview of Uses, Procedures, Applications, and Practice*. New York, New York: North-Holland, 1985.
- . *Handbook of Systems Analysis: Craft Issues and Procedural Choices*. New York, New York: North-Holland, 1988.
- Moder, Joseph J., Cecil R. Phillips, and Edward W. Davis. *Project Management With CPM, PERT, and Precedence Diagramming*. New York, New York: Van Nostrand Reinhold, 1983.
- Odiorne, George S. *MBO II: A System of Managerial Leadership for the 80s*. Belmont, California: Fearson Pitman Publishers, 1979.
- Olson, Timothy G., Watts S. Humphrey, and David H. Kitson. *Conducting SEI-Assisted Software Process Assessments*, CMU/SEI-89-TR-7 (also published as ESO-TR-89-09). Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute, 1989.
- Ould, Martyn. *Strategies for Software Engineering: Management of Risk and Quality*. Chichester, England: John Wiley & Sons, 1990.
- Putnam, L. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering* 4, 4:345-361, 1978.
- Putnam, L.H., and A. Fitzsimmons. Estimating Software Costs. *Datamation*, September 1979, 189-198, October 1979, 171-178, and November 1979, 137-140, 1979.
- RADC/COEE. *Software Top Level Design Document for the Software Life Cycle Support Environment (SLCSE)*. Griffith Air Force Base, New York: Rome Air Development Center, 1991.
- Royce, Winston W. "Managing the Development of Large Software Systems." In *Proceedings, IEEE WESCON*, 1970.
- Saaty, Thomas Lorie. *The Analytic Hierarchy Process*. New York, New York: McGraw-Hill International, 1980.
- . *The Analytic Hierarchic Process*. Pittsburgh, Pennsylvania: RWS Publications, 1990.

Schultz, H.P. *Software Management Metrics*, ESD-TR-88-001/M88-1. Bedford, Massachusetts: MITRE Corporation, 1988.

SEAS System Development Software Development: SSDM, Computer Sciences Corporation, July 1989.

Shuster, H. David. *Teaming for Quality Improvement: A Process for Innovation and Consensus*. Englewood Cliffs, New Jersey: Prentice-Hall, 1990.

*Software Engineering Handbook: General Electric Company*. New York, New York: McGraw-Hill, 1986.

Tausworthe, Robert C. The Work Breakdown Structure in Software Project Management. *The Journal of Systems and Software* (1980):181-86.

Thomsett, Michael C. *The Little Black Book of Business Statistics*. New York, New York: AMACOM, 1990.

Thomsett, Rob. *Third Wave Project Management*. Yourdon Press Computing Series. Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

TRW. *Process Model for High Performance Trusted Systems in Ada, Phase I Technical Report*. August 1989.

U.S. Air Force. *Configuration Management Practices for Systems, Equipment, Munitions, and Computer Software*, DOD-STD-483A. Washington, D.C.: U.S. Department of the Air Force, 1985.

U.S. Air Force Systems Command. *Software Management Indicators*. AFSCP 800-43. Washington, D.C.: U.S. Air Force Systems Command, 1986.

Ward, Paul, and Lloyd Williams. *The CASE Real-Time Method: An Object-Oriented Approach to Systems Engineering*. New York, New York: Dorset House, 1991.

Weber, Charles V., Mark C. Paulk, Cynthia J. Wise, James V. Withey. *Key Practices for the Capability Maturity Model*, CMU/SEI-91-TR-25 (also published as ESD-TR-91-25). Edited by Mary Beth Chrissis, Suzanne D. Couturiaux, and Ginny Redish. Pittsburgh, Pennsylvania: Carnegie Mellon University, Software Engineering Institute, 1991.

Weiss, D.M. *Evaluating Software Development by Analysis of Change Data*, TR-1120. College Park, Maryland: University of Maryland Computer Science Center, 1981.

Wild, Chris, Kurt Maly, and Lianfang Liu. Decision-Based Software Development. *Software Maintenance: Research and Practice* 3:17-43, 1991.

Wolff, Gerald J. The Management of Risk in System Development: "Project SP" and the "New Spiral Model." *Software Engineering Journal* (1989).

Wood, Bill. *A Guide to the Assessment of Software Development Methods*, CMU/SEI-88-TR-8 (also published as ESD-TR-88-009). Pittsburgh, Pennsylvania: Software Engineering Institute, 1988.

Zells, Lois. *Managing Software Projects*. Wellesley, Massachusetts: QED Information Sciences, Inc., 1990.

Zikmund, William G. *Business Research Methods*. Chicago, Illinois: The Dryden Press, 1988.

### Checklists

*Abbreviations for Use on Drawings, Specifications Standards, and in Technical Documents*, MIL-STD-12D, May 29, 1981.

Asaka, Dennis. *Review of Inspection Checklists*. 1991.

Birrell, N.D., and M.A. Ould. *A Practical Handbook for Software Development*. Cambridge, England: Cambridge University Press.

Department of Defense. *Military Handbook, Mission-Critical Computer Resources Software Support*, MIL-HDBK-347, May 22, 1990.

Fairley, Richard. *Software Project Management Course*. Software Engineering Management Association, 15-16, 15-18, 15-19.

———. *Software Project Risk Management*. Software Engineering Management Association, 1991.

Itabhi Corporation. *Successful Enterprise Review List* (in Table G-13, Early Evaluation of Project Items), 1991.

*Industrial Security Manual for Safeguarding Classified Information*, Department of Defense. DOD-5220.22-M. January 1991.

Itabhi Corporation. *General Project Review Questions* (in Table G-14, Ongoing Project Review Questions), 1991.

Longbow Apache Software Development Procedures. *Longbow Apache Software Development Procedures*. Vol. 15, Inspection Procedure. March 19, 1990.

Marciniak, John J., and Donald J. Reifer. *Software Acquisition Management: Managing the Acquisition of Custom Software Systems*. New York, New York: John Wiley & Sons, 1990.

Merriam-Webster Inc. Webster's Third New International Dictionary. Springfield, Massachusetts: Merriam-Webster Inc., 1981.

National Aeronautics and Space Administration. *Manager's Handbook for Software Development*, SEL-84-101, Revision 1. National Aeronautics and Space Administration, November 1990.

Ould, Martyn A. *Strategies for Software Engineering: The Management of Risk and Quality*. New York, New York: John Wiley & Sons, 1990.

*Software Development Plan*. DI-MCCR-80030A. February 29, 1988.

Software Product Assurance, IT1 Test Plan Checklist. JPL, April 11, 1988. (Received from Dr. Richard E. Fairley, Software Engineering Management Association.)

Software Test and Evaluation Manual, DoD Software Test and Evaluation Manual, Volume I, *Guidelines for the Treatment of Software in Test and Evaluation Master Plans*. Office of the Director Defense Test and Evaluation, May 1985.

*Software Test Plan*. DI-MCCR-80014A. February 29, 1988.

*Specification Practices*, MIL-STD-490A, June 4, 1985.

Spencer, Richard H. *Planning Implementation and Control in Product Test and Assurance*. Englewood Cliffs, New Jersey: Prentice-Hall, 29-30, 60-61, 147-50.

U.S. Government Printing Office. *Style Manual*. Washington, D.C.: U.S. Government Printing Office, 1984.

Validation Laboratory, Review of Inspection Checklists. Herndon, Virginia: Software Productivity Consortium, 1991.

### **Inspections Method Related**

Bisant, D.B., and J.R. Lyle. A Two-Person Inspection Method to Improve Programming Productivity. *IEEE Transactions of Software Engineering* 15, 10 (October 1989):1294-1304.

Brothers, L., V. Sembugamoorthy, and M. Muller. "ICICLE: Groupware for Code Inspection." In *CSCW 90 Proceedings* October 1990.

Bush, M. "Formal Inspections—Do They Really Help." In *NSIA Sixth Annual Joint Conference on Software Quality and Productivity*. Viewgraphs. April 1990.

Dunn, R. *Software Defect Removal*. New York, New York: McGraw-Hill, 1984.

Dyer, M. "Verification Based Inspections." *Software Verification Workshop Notebook*. Herndon, Virginia: Software Productivity Consortium, August 1990.

Fagan, M.E. Advances in Software Inspections. *IEEE Transactions on Software Engineering*, SE-12, 7 (1986):744-51.

Freedman, D.P., and G.M. Weinberg. *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*. Boston, Massachusetts: Little, Brown Company, 1982.

Martin, J., and W.T. Tsai. N-Fold Inspection: A Requirements Analysis Technique. *Communications of the ACM*, 33, 2 (February 1990).

Mays, R.G. Applications of Defect Prevention in Software Development. *IEEE Journal on Selected Areas in Communication* 8, 2 (February 1990):164-68.

Mays, R.G., et al. Experiences With Defect Prevention. *IBM Systems Journal* 29, 1 (1990):4-32.

Parnas, D.L., and D.M. Weiss. "Active Design Reviews: Principles and Practices." *IEEE Proceedings of the 8th International Conference on Software Engineering*. London, England: Institute of Electrical and Electronics Engineers, August 1985, 132-36.

Tripp, L.L., W.F. Struck, and B.K. Pflug. "The Application of Multiple Team Inspections on a Safety-Critical Software Standard." Submitted for conference publication, February 1991.

Yourdon, E. *Structured Walkthroughs*. 4th ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.



*This page intentionally left blank.*

# INDEX

Activities, 2-8  
    artifacts, C-1  
    specifications, A-1, B-1  
Alternatives, A-5  
Approach, A-5

Constraints, A-5  
Context, A-1

Development planning, 4-5, A-1  
    commitment, A-22  
    planning, A-19  
    scheduling, A-19

Estimate of the situation, A-6  
    developing, A-6  
    updating, A-6  
Evolutionary spiral process model, 3-1  
    and process engineering, 3-5  
    conceptual spiral, 3-3

Life-cycle models, 4-11

Methods, 2-8

Objectives, A-5

Process assets, 2-6, 4-13  
    engineering, 4-1  
Process definition, 2-5, 2-9, 4-11  
    high-level, 2-3, 3-2, 3-3  
    low-level, 2-4  
    project, 2-10, 5-3

Process engineering, 5-1  
    concepts, 2-1, 2-5  
    issues, 2-2  
    process, 2-10  
Product change control, A-28  
Product development, A-1  
    activities, B-1  
    monitoring, 4-7, A-25  
    reviewing, 4-7, A-25  
    verification, 4-7, A-23  
Project process, 5-3  
    enacting, 5-10  
    improving, 5-11  
    instantiating, 5-7  
    tailoring, 5-7

Reviews  
    of alternatives, A-18  
    of context, A-8  
    of development process, A-25  
    of progress, A-29  
    of risk analysis, A-13  
    of technical product, 4-8, A-27  
Risk analysis, 4-4, A-1, A-10  
Risk aversion  
    commitment, A-15  
    execution, 4-5, A-17  
    planning, 4-4, A-14

Software Engineering Institute CMM, 4-3  
Spiral planning, commitment, A-32  
Spiral planning documents, A-3  
    updating, A-30  
Stakeholders, A-5

Tactical plan, 4-2  
    updating, 4-2, 4-12

*This page intentionally left blank.*